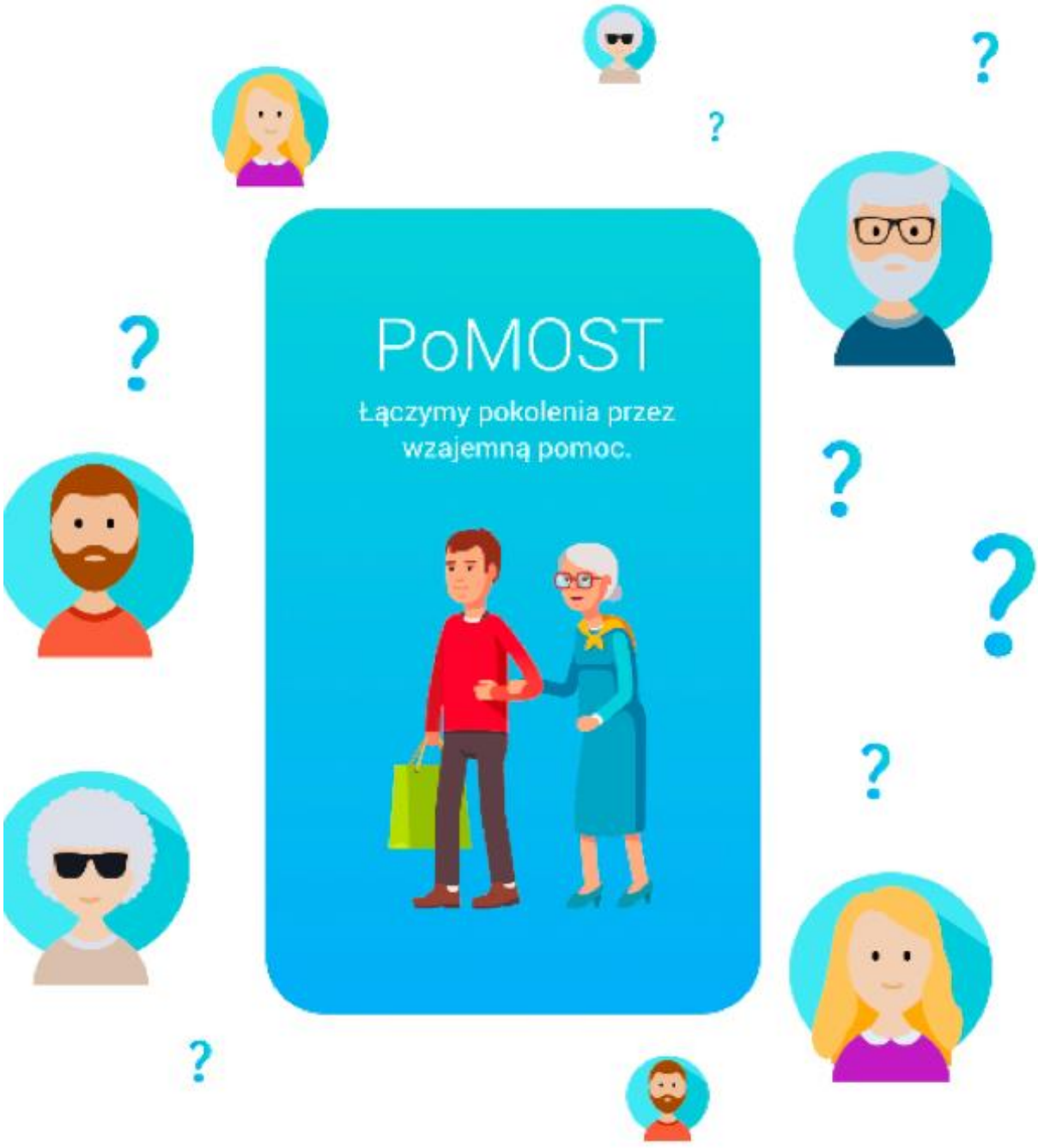


Raport z testu aplikacji poMOST

Kraków 2018

SPIS TREŚCI

Opis narzędzia – aplikacja poMOST

Z wiekiem problemy zdrowotne intensyfikują się. Osoby starsze, ale także szeroka kategoria osób o różnym stopniu niepełnosprawności, mają specyficzne potrzeby, które powinny być realizowane w środowisku lokalnym z uwzględnieniem wspierającego charakteru lokalnych i rodzinnych sieci społecznych. Badania pokazują bowiem, że osoby starsze chcą jak najdłużej pozostać niezależne i żyć we własnym mieszkaniu. Preferencja ta nie jest związana z dominującym w danym państwie rozwiązaniem systemowym w zakresie opieki nad osobami starszymi. Zatem zarówno w Norwegii (z dominującym instytucjonalnym wsparciem), jak i w Polsce (gdzie za opiekę nad osobami starszymi odpowiedzialna jest rodzina), dążenie do utrzymania autonomii przez osoby starsze jest powszechnym zjawiskiem. Rozwiązania takie umożliwiają nie tylko niezależne życie, ale również poczucie przynależności społecznej i sprawstwa. Potrzeby osób starszych w zakresie opieki długoterminowej mogą być w Polsce w coraz mniejszym stopniu zaspokajane w sposób tradycyjny, czyli poprzez nieformalne wsparcie rodzinne. Luka pomiędzy coraz większymi potrzebami zwiększającej się liczebnie kategorii osób starszych, a możliwościami zaspokajania ich przez młodsze generacje może być wypełniona przez rozwiązania technologiczne. Pojawia się coraz większa liczba programów skierowanych na stworzenie innowacyjnych modeli wsparcia niezależnego życia osób starszych. Odwołują się one do technologii wpierających (*assistive&ambient technology*) osoby w podeszłym wieku w wykonywaniu codziennych czynności oraz pozwalają na szybkie reagowania rodziny oraz służb medycznych w oparciu o docierające z otoczenia osoby starszej niepokojące sygnały. Umożliwiają one również dostęp do dóbr, usług i miejsc przyczyniając się tym samym do integracji społecznej osób starszych. Technologie wspierające, to szeroki wachlarz urządzeń. Mogą one przybierać postać od prostych zawieszek czy też bransoletek z alarmem, poprzez systemy czujników, aplikacje, panele monitorujące przestrzeń mieszkalną osoby starszej, sensory zintegrowane z odzieżą (*wearable technologies*), po roboty. Również w przypadku młodszych generacji technologie te mogą być pomocne – chociażby poprzez umożliwienie połączenia pracy zawodowej i zobowiązań opiekuńczych rodziny - zwłaszcza kobiet, które w Polsce są kulturowo zobligowane do opieki nad starszymi rodzicami. Rozwiązania takie rozszerzają zatem definicję sieci wspierających, jednak równocześnie mogą prowadzić do izolacji społecznej, przekształcać relacje społeczne w nieznanym kierunku oraz redefiniować poczucie intymności.

Aplikacja międzygeneracyjna poMOST odpowiada na zdiagnozowana w ramach wcześniejszych badań własnych potrzeby starzejącego się społeczeństwa polskiego. W Unii Europejskiej proporcja osób 65 lat i więcej zwiększy się z 18% w 2013 roku do 28% w 2060 roku. W Polsce będzie to wzrost z 14% do 33%. Zwiększy się również w UE (ale i w Polsce) grupa osób 80 lat i więcej, która w największym stopniu wymaga opieki: z 5% w 2013 roku do 12% w 2060 roku. W ślad za zmianami demograficznymi, modyfikacji ulegnie również współczynnik całkowitego obciążenia demograficznego: z 42 osób w wieku nieprodukcyjnym przypadających na 100 osób w wieku produkcyjnym w 2010 roku do 58 osób w roku 2035 (GUS 2014). W państwach, takich jak Polska, gdzie dominuje rodzinny model opieki nad osobami starszymi, pojawi się problem związany z brakiem zasobów opiekuńczych. Liczba osób starszych wymagających opieki będzie rosnąć, a liczba osób młodszych mogących opiekę zapewnić, będzie maleć. Obrazuje to wskaźnik potencjału pielęgnacyjnego – czyli stosunek liczby kobiet w wieku 45-64 lat do osób w wieku 80 i więcej - który w samej Małopolsce spadnie z 325 potencjalnych opiekunek w 2013 do prognozowanych 128 w 2050.

Na wspomniane procesy demograficzne nakładają się zmiany w zakresie organizacji pracy i migracji (krajowej i zagranicznej) związanej z jej poszukiwaniem. Jak wynika z badań własnych (*Migranci i ich starzejący się rodzice. Transnarodowy system opieki międzygeneracyjnej*), odległość geograficzna nie likwiduje zobowiązań opiekuńczych dorosłych dzieci wobec starszych rodziców, lecz je modyfikuje poprzez konieczność włączenia dalszej rodziny, przyjaciół oraz korzystania z nowych technologii, które ułatwiają prowadzenie niezależnego życia przez osoby starsze, których główni opiekunowie nie mieszkają w bliskiej odległości.

Tymczasem jak wynika z danych Głównego Urzędu Statystycznego, większość gospodarstw domowych seniorów w Polsce, to gospodarstwa jednoosobowe prowadzone najczęściej przez kobiety, które statystycznie żyją dłużej od mężczyzn. Co więcej, w grupie osób 70+ aż co druga osoba zgłasza problem z przejściem samodzielnie odcinka dłuższego niż 500 metrów. W odniesieniu do kwestii możliwości prowadzenia gospodarstwa domowego można wnioskować, że występujące ograniczenia zdrowotne nie pozwalają osobom starszym wykonywać okazjonalnie ciężkich prac domowych. Co druga osoba w wieku 65+ deklaruje trudności z takimi czynnościami. Także robienie codziennych zakupów sprawia trudność co trzeciej starszej osobie, a w dalszej kolejności wykonywanie lżejszych prac domowych czy też dbanie o sprawy administracyjne czy finansowe. Możliwość wykonywania samodzielnie tych czynności maleje wraz z wiekiem. Prawie 28% starszych osób zgłaszających ograniczenia w prowadzeniu gospodarstwa domowego nie ma żadnej pomocy i musi sobie radzić sama (GUS 2016: Ludność w wieku 60 lat i więcej). Nawet jeżeli osoby starsze otrzymują pomoc w postaci dorosłych dzieci mieszkających w pobliżu, to ich odciążenie poprzez wykorzystanie lokalnego potencjału społecznego w postaci drobnej pomocy sąsiedzkiej aranżowanej poprzez aplikację poMOST, jest przez badane osoby starsze pozytywnie oceniane. Odciążenie głównych opiekunów – przede wszystkim kobiet - wpływa bowiem pozytywnie na ich stan zdrowia, stabilność emocjonalną, możliwość łączenia obowiązków zawodowych i opiekuńczych. Nie chodzi bowiem o to, aby społeczność lokalna zastąpiła opiekunów (instytucjonalnych czy też rodzinnych), lecz o to, aby ich wesprzeć poprzez drobne, sąsiedzkie dobre uczynki.

Proponowane narzędzie, bazujące na internetowych technologiach komunikacyjnych, pozwoli na prowadzenie przez osoby starsze niezależnego życia oraz integrację ze lokalną społecznością sąsiedzką. poMOST ułatwi kontakt między potrzebującą pomocy osobą starszą, a lokalną społecznością w postaci dobrych uczynków, takich jak: zrobienia zakupów, podwiezienia np. do lekarza lub na spotkanie z przyjaciółmi, pomocy w znalezieniu fachowca, odebrania paczki z poczty, zrealizowanie recepty itp. Lista funkcjonalności została określona w trakcie badań *user experience* przy zaangażowaniu osób starszych testujących aplikację. Badania te pokazują również, iż poprzez zaangażowanie lokalnej społeczności sąsiedzkiej w pomoc osobom starszym, zwiększeniu może ulec nie tylko poczucie bezpieczeństwa w zakresie podstawowych funkcji życiowych, ale również integracja społeczności lokalnej poprzez zaangażowanie mieszkańców, przyjaciół i krewnych osoby starszej. Wygenerowany w procesie lokalnej pomocy kapitał społeczny pozytywnie wpłynie na jakość życia nie tylko osób starszych, lecz całej społeczności. Pomoc sąsiedzka znajduje się obecnie – zdaniem badanych osób starszych – w stanie erozji w porównaniu do przeszłości. Wśród czynników wpływających na osłabienie więzi sąsiedzkich badani wskazywali przede wszystkim na zwiększający się odsetek mieszkań przeznaczonych na wynajem krótkoterminowy, co skutkuje zwiększeniem się młodych lokatorów - bardzo często studentów - na okres poniżej jednego roku. Biorąc pod uwagę powszechność wykorzystywania aplikacji mobilnych przez osoby młodsze oraz rosnący udział osób starszych w korzystaniu z technologii internetowych. Jak wynika z Diagnozy Społecznej oraz licznych badań realizowanych na przykład przez CBOS, około 15% osób w wieku 65+ korzystało z Internetu w

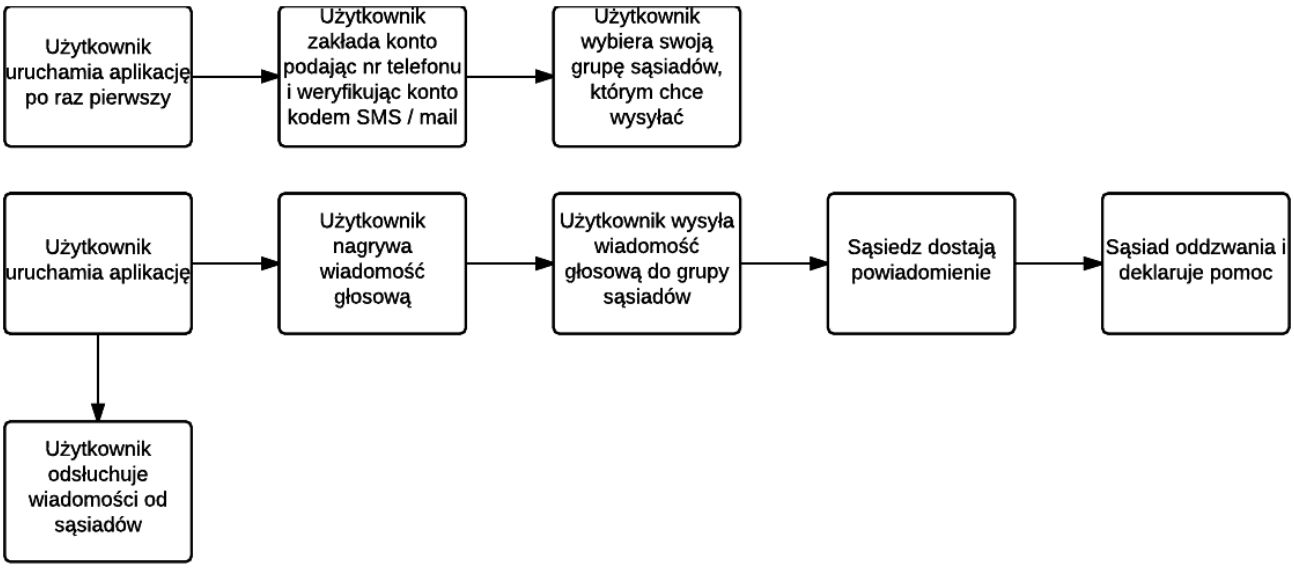
2015 roku. Niemniej warto zwrócić uwagę na dwa fakty. Po pierwsze, z roku na rok liczba użytkowników w starszych kategoriach wieku rośnie. Wynika to nie tylko z licznej oferty edukacyjnej kierowanej do osób starszych (np. w ramach Uniwersytetów Trzeciego Wieku), lecz również z tego, że młodsze kategorie stopniowo wchodzą w wiek starszy już z wiedzą i umiejętnościami technologicznymi.

Istota innowacji aplikacji poMOST

Aplikacja poMOST łączy osoby starsze z osobami ze środowiska lokalnego (sąsiadami, wolontariuszami lub usługodawcami) mogącymi udzielić wsparcia w realizacji codziennych czynności. Poprzez użycie aplikacji użytkownik potrzebujący wsparcia w drobnych czynnościach (takich jak zrobienie zakupów, podwiezienie do lekarza, wykupienie recepty, odebranie, nadanie paczki, wezwania pomocy w nagłym przypadku) będzie miał możliwość szybkiego skontaktowania się z osobami przebywającymi teraz lub w przyszłości w jego okolicy. Zgłoszenie wyświetli się pobliskim użytkownikom, którzy w ramach drobnej przysługi lub działając „przy okazji" będą w stanie odpowiedzieć na daną potrzebę. We wdrożeniu aplikacji udział będą brały także najbliższe osoby, które mogą wytłumaczyć działanie aplikacji osobie potrzebującej pomocy. Ze względu na fakt, iż aplikacja ma swoje zakorzenienie w środowisku lokalnym i jest oddolnie tworzona, ważna jest możliwość dodania zdjęcia oraz danych kontaktowych do profilu, co pozytywnie wpływa na poziom zaufania, a co za tym idzie na sukces wdrożeniowy innowacji. Sam interfejs został zaprojektowany zgodnie z wytycznymi koncepcji *user friendly*, zgodnie z którą prostota rozwiązań łączy się z efektywnością aplikacji.

Poniższy schemat przedstawia proces konfiguracji i użytkowania aplikacji:

Schemat 1. *User Journey* konceptu nowej aplikacji



Wady i zalety rozwiązania

Opracowana innowacja posiada swoje wady i zalety. Warto podkreślić, iż wiele elementów aplikacji jest dostosowanych do potrzeb i możliwości użytkowników, dla których korzystanie z internetowych rozwiązań komunikacyjnych może być problematyczne. Zatem niektóre wady aplikacji należy rozpatrywać w kontekście specyfiki użytkowników końcowych, dla których aplikacja została przygotowana. Zwiększenie funkcjonalności aplikacji, a co za tym często idzie stopnia jej skomplikowania, może bowiem wpłynąć negatywnie na liczbę aktywnych użytkowników-osób potrzebujących pomocy. Należy pamiętać, że żadnej aplikacji mobilnej nie można rozpatrywać jako innowacyjnej w skali kraju, a tym bardziej w skali świata. Jako innowację należy rozpatrywać zespół funkcjonalności specjalnie zaprojektowanych pod kątem konkretnej grupy użytkowników końcowych, w tym tych jej członków, którzy są najbardziej wymagający i dla których stopień skomplikowania zniechęca do korzystania z aplikacji.

- **Zalety aplikacji poMOST**

1. Prosty, czytelny i intuicyjny interfejs aplikacji opierający się na zminimalizowanej liczbie czynności koniecznych do wysłania prośby o pomoc oraz zareagowania na nią.
2. Dyskrecja działania. Badania testowe pokazały, że stosunek do nowych technologii związanych z opieką zależy w dużej mierze od tego, czy jest ona „widoczna" i wskazująca na problemy użytkownika. Aplikacja poMOST nie różni się od innych aplikacji, w dowolnym momencie może zostać też wyłączona.
3. Proste algorytmy definiujące rodzaj pomocy. Użytkownik potrzebujący pomocy może wybrać formę pomocy spośród kilku predefiniowanych obszarów (np.: zakupy, poczta, transport) i wysłać predefiniowaną prośbę o pomoc do osób znajdujących się w pobliżu. System komunikacji i nawigacji po systemie jest zatem uproszczony i nie wymaga dokładnego opisywania problemu, co może być trudne dla starszych użytkowników aplikacji.
4. Aplikacja została oddolnie zaprojektowana i działająca na zasadzie oddolnej pomocy społecznościowej. Proponowane narzędzie, bazujące na internetowych technologiach komunikacyjnych, pozwala na prowadzenie przez osoby starsze niezależnego życia oraz integrację z lokalną społecznością sąsiedzką. Aplikacja pozytywnie wpływa na jakość życia osób starszych i ograniczonych mobilnie, może być wykorzystywana zarówno przez małe lokalne społeczności, jak i większe instytucje świadczące usługi opiekuńcze czy doradcze.
5. Rozwiązanie opierające na się na bezpłatnej, drobnej pomocy realizowanej „przy okazji" zwiększa zaangażowanie wolontariuszy w pomoc osobom starszym.

- **Wady aplikacji poMOST**

1. Działanie aplikacji opiera się na rozwiązaniach internetowych i mobilnych, co może stanowić problem dla starszych użytkowników aplikacji. Co prawda badania pokazują, że coraz więcej osób starszych korzysta z technologii mobilnych w codziennym życiu, niemniej przekonanie użytkowników do korzystania z aplikacji może być trudne ze względu na jej innowacyjny charakter. Metodologia wdrażania innowacji zakłada, że tak zwani „wcześni innowatorzy", to nie więcej niż 5% potencjalnych użytkowników końcowych. Ludzie niechętnie korzystają z innowacji, o których wcześniej nie słyszeli a „wczesnych innowatorów" jest stosunkowo niewielu. Pamiętać jednak należy, że to innowacyjna aplikacja i w wersji testowej będzie używana przez

„wczesnych innowatorów” – osoby, które są zainteresowane nowymi technologiami i swobodnie poruszają się w przestrzeni internetowej. Osoby takie pełnić będą funkcje liderów opinii i na dalszych etapach wdrażania innowacji mogą brać udział w akcjach promocyjnych, dyskusjach z osobami starszymi, w celu przekonania ich o pozytywnych aspektach korzystania z aplikacji.

2. Działanie aplikacji opiera się na idei oddolnej pomocy bazującej na zaufaniu społeczny. Jest to źródło ryzyka, ponieważ Polskę cechuje niski poziom zaufania społecznego. Wyzwaniem dla instytucji zarządzającej będzie przekonanie osób starszych oraz potencjalnych osób pomagających, do współpracy w ramach lokalnej społeczności. Jak pokazują badania, Polacy pomagają najchętniej członkom własnej rodziny i w niewielkim stopniu angażują się aktywnie w wolontariat. Niemniej projekt zakłada współdziałanie osób, które mieszkają blisko siebie a więc mogą się znać z widzenia. Warto pamiętać, że zniwelowania tego ryzyka pozytywnie wpłynie na poziom kapitału społecznego i zaufania wobec współmieszkańców.

3. Aplikacja – jako innowacyjne narzędzie o charakterze prototypu - działa jedynie na smartfonach z systemem Android.
4. Uproszczony system komunikacji między wolontariuszem a osobą potrzebującą wsparcia stanowi zarówno zaletę, jak i wadę. Nieco bardziej skomplikowane algorytmy (np. lokalizacyjne) pozwoliłyby bowiem na określenie, w jakim obszarze znajdują się osoby potrzebujące wsparcia oraz wybranie konkretnej osoby. Jednak znytnie skomplikowanie aplikacji na początkowym etapie wdrożenia może odstraszać użytkowników.
5. Wysłanie zapytania o pomoc jest aktualnie uproszczone i dla bardziej zaawansowanych użytkowników mogłyby powstać algorytmy uszczegóławiające rodzaj pomocy, aby odbiorca/pomocnik mógł otrzymać dokładną informację na temat formy pomocy (np. w przypadku drobnych zakupów mogłoby to być doprecyzowanie o jaki dokładnie rodzaj produktów chodzi, w jakim sklepie zakupy powinny być zrobione, w jakim przedziale czasowym winny być dostarczone, jaki jest budżet).
6. Aktualna wersja aplikacji zawiera także uproszczone algorytmy rozpoznawania mowy i poleceń, a użytkownik, w szczególności starszy, mógłby wysłać prośbę używając jedynie komunikatów głosowych.
7. Rozwiązanie nie posiada funkcjonalności wysyłania wiadomości między użytkownikami, która mogłaby uprościć kontakt między nimi i nie będzie wymagała kontaktu telefonicznego w celu doprecyzowania prośby. Niemniej wysyłanie wiadomości jest dla wielu osób starszych trudniejsze od przeprowadzenia rozmowy telefonicznej w konkretnej sprawie.

8. Brak systemu weryfikującego użytkowników w celu zwiększenia bezpieczeństwa. Taki system powinien być zaimplementowany przez instytucje zarządzające siecią wsparcia w ramach aplikacji poMOST.

Wnioski płynące z badań wstępnych (diagnostycznych)

Aplikacja poMOST powstała w oparciu o przeprowadzone wcześniej badania diagnostyczne w postaci dwóch wywiadów grupowych z przedstawicielami użytkowników końcowych – osobami starszymi korzystającymi z internetowych urządzań mobilnych. Podczas analizy badań udało się wyodrębnić trzy persony mieszczące się w grupie docelowej oraz wyciągnąć trzy najważniejsze wnioski, mające znaczący wpływ na projekt: a) uczestnicy procesu pomocy sąsiedzkiej; b) penetracja smartfonów w grupie docelowej; c) najbardziej atrakcyjna forma kontaktu z sąsiadami i rodziną.

Pierwszy ważnym spostrzeżeniem z badań, które ma istotne znaczenie dla projektu, są osoby uczestniczące w pomocy sąsiedzkiej. Podczas wywiadów uczestnicy wskazywali, że najczęściej zarówno oni, jak i im, pomagają osoby z tego samego pokolenia. Seniorzy wskazywali, że nie utrzymują w większości kontaktów z młodszymi pokoleniami sąsiadów z powodu kilku czynników: a) zbyt duża rotacja sąsiadów w budynku mieszkalnym; b) młodsze pokolenia są mniej obecne w mieszkaniach i na osiedlach, a spędzają aktywnie czas poza domem; c) korzystają z technologii oraz utrzymują więzi społeczne dzięki temu z osobami mieszkającymi nie tylko w najbliższym sąsiedztwie.

Uczestnicy badania, mówiąc o swoich doświadczeniach związanych z pomocą sąsiedzką, bardzo często porównywali istnienie zjawiska na przełomie 30 - 40 lat, wskazując na różnice w intensywności więzi społecznych wcześniej, a dziś. Podkreślali również, jakie czynniki w ich odczuciu mają wpływ na ich osłabienie, za jeden z głównych wskazując zwiększający się odsetek mieszkań przeznaczonych na wynajem krótkoterminowy, co skutkuje zwiększeniem się młodych lokatorów (bardzo często studentów) na okres poniżej jednego roku. Większość uczestników podczas badań wskazywała na dużo łatwiejsze budowanie więzi z innymi osobami z tego samego pokolenia, zwłaszcza jeśli są to osoby przebywające w jednym miejscu przez długi okres czasu (powyżej 10 lat - podczas badań była to dolna granica).

Zdaniem przedstawicieli grupy docelowej, wśród głównych problemów zjawiska pomocy sąsiedzkiej dziś należy wymienić: a) bardzo wysoka rotacja mieszkańców w budynkach - coraz większy odsetek mieszkań jest wynajmowany; b) nieznajomość sąsiadów - uczestnicy porównywali, że więzi społeczne wcześniej (ok. 20 - 30 lat temu - przyp. wypowiedzi uczestników) były znacznie bliższe, a sąsiedzi utrzymywali bliskie, często określane mianem "rodzinnych", relacje; c) uszczuplanie się liczby sąsiadów tego samego pokolenia - uczestnicy badania wskazywali, że dużo łatwiej nawiązać im relacje z przedstawicielami tego samego pokolenia i zwracali uwagę na problem zmiany środowiska sąsiedzkiego w przypadku wyprowadzenia się jednego członka społeczności, jego odejścia czy ewentualnej śmierci.

Za czynniki ważne, mające wpływ na budowanie bliższych więzi społecznych, a co za tym idzie zwiększenie sposobności do udzielania i przyjmowania pomocy sąsiedzkiej, grupa odbiorców uważała: a) staż - jak długo inna osoba mieszkała w sąsiedztwie; b) częstotliwość przypadkowych spotkań - np. w windzie, na korytarzu, w trakcie spacerów; c) oraz to samo piętro - uczestnicy obu wywiadów niezależnie wskazywali, że utrzymywali w przeszłości i teraz bliższe stosunki z sąsiadami, którzy przebywają na tym samym piętrze, podkreślając często, że osoby z innych pięter (w przypadku dużych kompleksów mieszkalnych) czy klatek schodowych traktują z większą rezerwą.
Zaleceniem płynącym z powyższej obserwacji jest, by aplikacja była skierowana do osób starszych - zarówno po stronie osoby udzielającej pomocy jak i beneficjenta.

Ostatnim aspektem, mającym duży wpływ na kształt aplikacji tworzonej w ramach projektu, jest preferowana forma kontaktu przez osoby starsze. Seniorzy w większości podczas badań preferowali kontakt osobisty bądź połączenia głosowe, rzadko wskazując na wykorzystanie SMSów, czy komunikatorów typu Skype, jednostkowo wspominając o poczcie elektronicznej. Aplikacja opiera się zatem na predefiniowanych formach wsparcia, które wybiera się z listy form pomocy bez konieczności pisania wiadomości tekstowych.

Test aplikacji

Podjęte działania

Proces przygotowania i testowania innowacji przebiegał w okresie sześciu miesięcy i składał się z kilku etapów.

Październik – grudzień 2017:

1. Na podstawie rekomendacji z badań diagnostycznych została przygotowana aplikacja działająca na urządzeniach mobilnych z systemem Android.
2. W oparciu o badania diagnostyczne zweryfikowano interfejs aplikacji tak, aby elementy graficzne oraz system nawigacji po aplikacji były czytelne i intuicyjne dla starszych użytkowników.
3. Przygotowano instrukcję obsługi aplikacji.
4. Przygotowana według wniosków z badań aplikacja została przygotowana do dwumiesięcznych testów w środowisku lokalnym.

Grudzień 2017

1. Przeprowadzono rekrutację użytkowników, którzy przyjęli rolę wolontariuszy pomagających osobom starszym.
2. Przeprowadzono szkolenie wolontariuszy, którzy zdobyli wiedzę na temat komunikacji z osobami starszymi oraz radzenia sobie z potencjalnymi problemami wyłaniającymi się w procesie korzystania z aplikacji.
3. Wolontariusze rozpoczęli proces rekrutacji potencjalnych starszych użytkowników końcowych.

Styczeń – marzec 2018

1. Użytkownicy zainstalowali aplikację na urządzeniach mobilnych i korzystali z niej w miarę własnych potrzeb.
2. W przypadku funkcjonalności „Alarm” przeprowadzono test weryfikujący poprawne działanie tej funkcji.
3. Przeprowadzono badanie kontekstowe wśród osób testujących aplikację określające działania rutynowe, problemy, istniejące sieci wsparcia oraz możliwe uzupełnienia w funkcjonalnościach aplikacji.
4. Użytkownicy aplikacji prowadzili dziennik z zapisem problemów pojawiających się w trakcie korzystania z aplikacji.
5. Wolontariusze na bieżąco zapisywali swoje obserwacje.
6. W ostatnim tygodniu testów wolontariusze przeprowadzili ze starszymi użytkownikami aplikacji test użyteczności z wykorzystaniem kwestionariusza ankiety.
7. Na podstawie wniosków płynących z testów na bieżąco modyfikowano aplikację. Działania modyfikujące dotyczyły głównie interfejsu aplikacji.
8. Na podstawie zgromadzonych w ramach testów danych przygotowano niniejszy raport.

Cel testowania aplikacji poMOST

Głównym celem testu aplikacji było sprawdzenie funkcjonalności wśród potencjalnych użytkowników końcowych. Rezultaty testu pozwoliły zdefiniować ostateczną listę funkcjonalności, wyeliminować błędy techniczne oraz zmodyfikować element graficzne aplikacji. Wyniki testu pozwoliły na opracowanie ostatecznej wersji aplikacji poMOST dostosowanej do możliwości poznawczych oraz umiejętności technologicznych użytkowników końcowych.

Cele testu:

- Opis kontekstu, w jakim użytkownicy końcowi korzystają z aplikacji;
- Określenie rutyny dnia codziennego, która definiuje systematycznie pojawiające się problemy osób starszych;
- Ulokowanie aplikacji w kontekście innych sposobów zapewniania przez użytkowników końcowych bezpiecznego i samodzielnego funkcjonowania w przestrzeni domu
- Zdefiniowanie ostatecznej listy funkcjonalności w oparciu o obserwację, wywiady oraz prowadzone dzienniki
- Wykrycie nieprawidłowości technicznych
- Opracowanie ostatecznej wersji aplikacji poMOST dostosowanej do umiejętności technologicznych osób starszych

Metodologia badań testujących aplikację poMOST

Zastosowana metodologia opiera się na założeniach *user experience* z wykorzystaniem zarówno jakościowych, jak i ilościowych technik badawczych. Osoby przeprowadzające test realizowały badanie z użytkownikami końcowymi w czasie licznych wizyt, w ramach których obserwowały sposoby rozwiązywania problemów, przeprowadzały wywiad oraz zadawały zestandaryzowane pytania dotyczące użyteczności aplikacji. Badanie testowe miało zatem charakter etnograficzny zawierający pogłębione i całościowe ujęcie kwestii niezależnego funkcjonowania w przestrzeni domu oraz sposobów radzenia sobie z problemami dnia codziennego. Z jednej strony takie podejście pozwoliło na zdiagnozowanie problemów, zrozumienie oczekiwań użytkowników końcowych, poznanie kontekstu społecznego, w którym funkcjonują oraz zdefiniowanie roli nowych technologii w opiece nad osobami starszymi. Z drugiej strony, zastosowana metodologia umożliwiła zidentyfikowanie problemów technicznych oraz pozwoliła na doprecyzowanie takich elementów aplikacji jak: kolorystyka, czcionka, łatwość rejestracji oraz nawigacji po poszczególnych elementach aplikacji.

Test aplikacji opierał się na dwóch powiązanych ze sobą części:

- **Badanie kontekstowe**

  W tej części badania diagnozie została poddana rutyna dnia codziennego; sieci społecznych potencjalnych użytkowników końcowych, stosunek do nowych technologii oraz określenia kompetencji technologicznych. Badanie to opierało się na jakościowym wywiadzie z elementami zestandaryzowanych pytań kwestionariuszowych oraz analizy sieci społecznych.

- **Test użyteczności aplikacji**

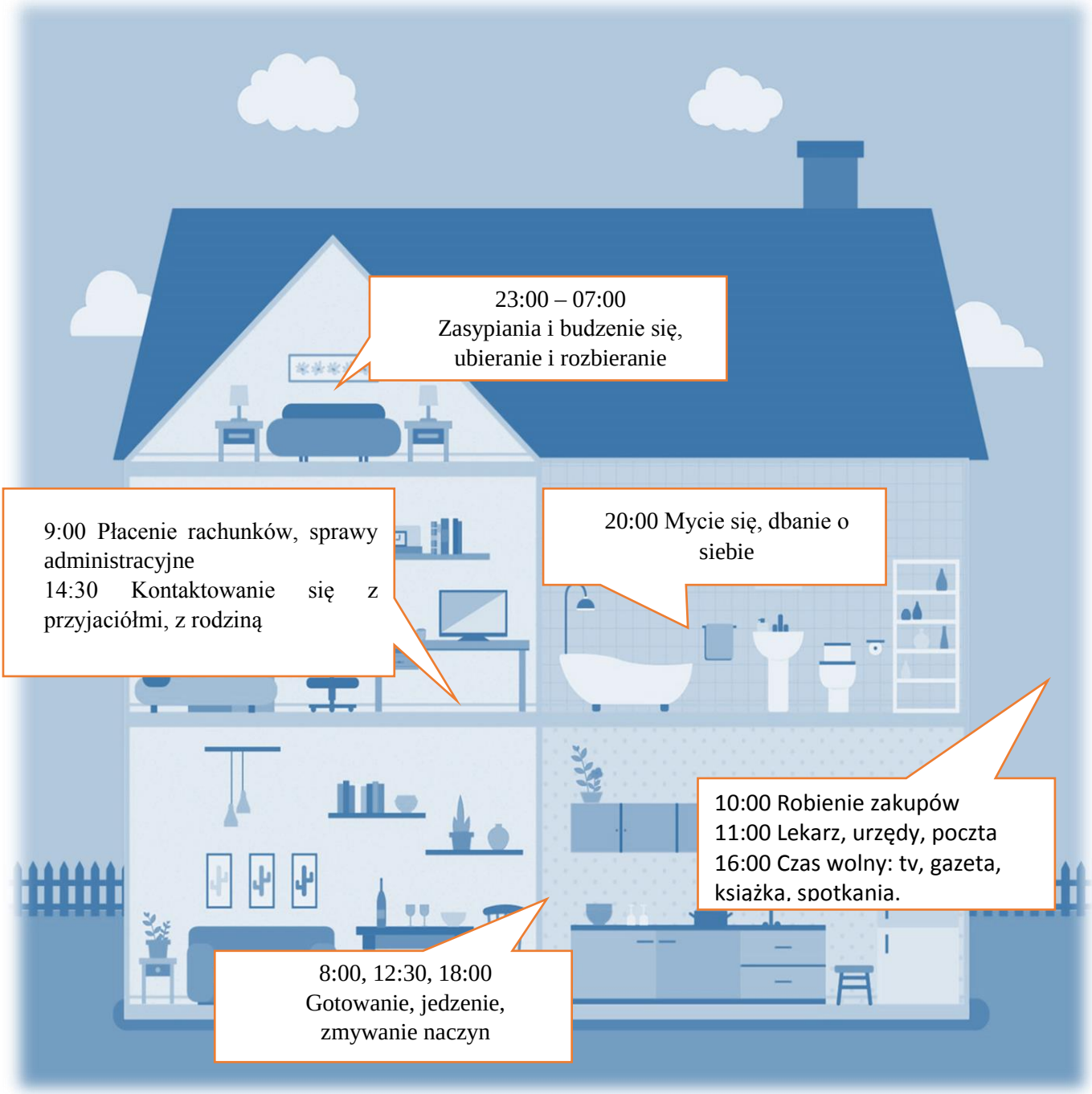
  W tej części miała miejsce weryfikacja użyteczności testowanej aplikacji oraz identyfikacja problemów technicznych. Badanie to realizowane było z wykorzystaniem kwestionariusza ankiety oraz dziennika, w którym osoby testujące zapisywały pojawiające się w czasie testu problemy.

Zgodnie ze specyfikacją innowacji, w teście wzięło udział dziesięć osób: zarówno osób testujących, jak i osób udzielających wsparcia. Wiek badanych starszych użytkowników zawierał się w przedziale 71-78 lat. Większość badanych mieszka w jednorodzinnych mieszkaniach w bloku z windą. W jednym przypadku mieszkanie osoby starszej znajdował się na 3 piętrze w bloku bez windy. Badani użytkownicy końcowi znajdują się w dobrym stanie zdrowia i w wykonują większość czynności domowych – takich jak sprzątanie, gotowanie, prasowanie – samodzielnie.

Prezentacja wyników. Badanie kontekstowe

Rutyna dnia codziennego

Badani mieli możliwość określenia swoich codziennych czynności rutynowych. Poniższy schemat przedstawia owe czynności w podziale na różne pomieszczenia, czas ich wykonywania.



Schemat 2. Uogólniony model rutynowych czynności.
Ilustracja pochodzi z: Freepik.com http://www.freepik.com. Licencja Creative Commons.

Rutyna dnia codziennego. Problemy osób starszych – potencjalne uzupełnienie aplikacji pomost

Prezentowane tabele opisują główne trudności i problemy funkcjonalne w codziennym życiu w przestrzeni domowej. Były one wymieniane przez badanych jako potencjalne dodatkowe funkcjonalności aplikacji poMOST. Niemniej, ze względu na architektoniczny charakter owych trudności, uwzględnienie ich w aplikacji nie było możliwe. Wyjątkiem jest funkcja alarmu w przypadku upadku.

Stopień ważności (priorytet), zdefiniowany przez badacza, to: niewielki, średni oraz duży. Biorąc pod uwagę fakt, iż badania miały charakter jakościowy a nie ilościowy, częstość nie jest głównym czynnikiem świadczącym o ważności.

- Niewielki stopień ważności: dotyczy tych trudności, które odzwierciedlają drobne przeszkody w codziennym funkcjonowaniu I dotyczą głównie estetyki czy też większego komfortu.
- Średni stopień ważności: to potrzeba lub przeszkoda, które utrudniają codzienne funkcjonowanie, wywołują poczucie strachu i dyskomfort.
- Wysoki stopień ważności: dotyczy priorytetowych potrzeb, które nie mogą być w pełni zaspokojone, wywołują ból oraz stanowią zagrożenie dla zdrowia i życia.

- **Czynności związane ze kładzeniem się do łóżka, spaniem oraz budzeniem się**

Problem	Życzenia/propozycje rozwiązania	Priorytet
Łóżko	Urządzenie do kontroli wysokości łóżka	Wysoki
Kontrola światła	Automatyczna regulacja oświetlenia pokoju; automatyczne włączenie/wyłączenie światła pod wpływem ruch/braku ruchu;	Średni

- **Higiena osobista**

Problem	Życzenia/propozycje rozwiązania	Priorytet
Branie prysznica – groźba upadku	Maty antypoślizgowe; uchwyty; możliwość wzięcia prysznica w pozycji siedzącej	Wysoki
Czystość łazienki	Automatyczna spłuczka	Niewielki
Potrzeba wyższego komfortu brania prysznica	Automatyczny dozownik środków czystości	Niewielki

- **Zarządzanie gospodarstem domowym**

Problem	Życzenia/propozycje rozwiązania	Priorytet
Częste gubienie przedmiotów	System lokalizacji przedmiotów	Wysoki
Ból przy myciu podłóg	Automat do mycia podłóg, pomoc zewnętrzna	Wysoki
Problem z obsługą pralki	Prostszy interfejs; umieszczenie pralki wyżej	Niewielki
Niekomfortowa wysokość stołu	Możliwość zmiany wysykości stołu	Niewielki

- **Aktywności poza domem**

Problem	Życzenia/propozycje rozwiązania	Priorytet
Strach przed upadkiem	System powiadomień sąsiadów, rodziny o wypadku	Wysoki

- **Dokumenty, zarządzanie budżetem**

Problem	Życzenia/propozycje rozwiązania	Priorytet
Zapominanie o płaceniu rachunków	System powiadomień	Wysoki

- **Aktywność społeczna**

Problem	Życzenia/propozycje rozwiązania	Priorytet
Problem ze słuchem	Automatoczne dopasowanie dźwięku	Średnie

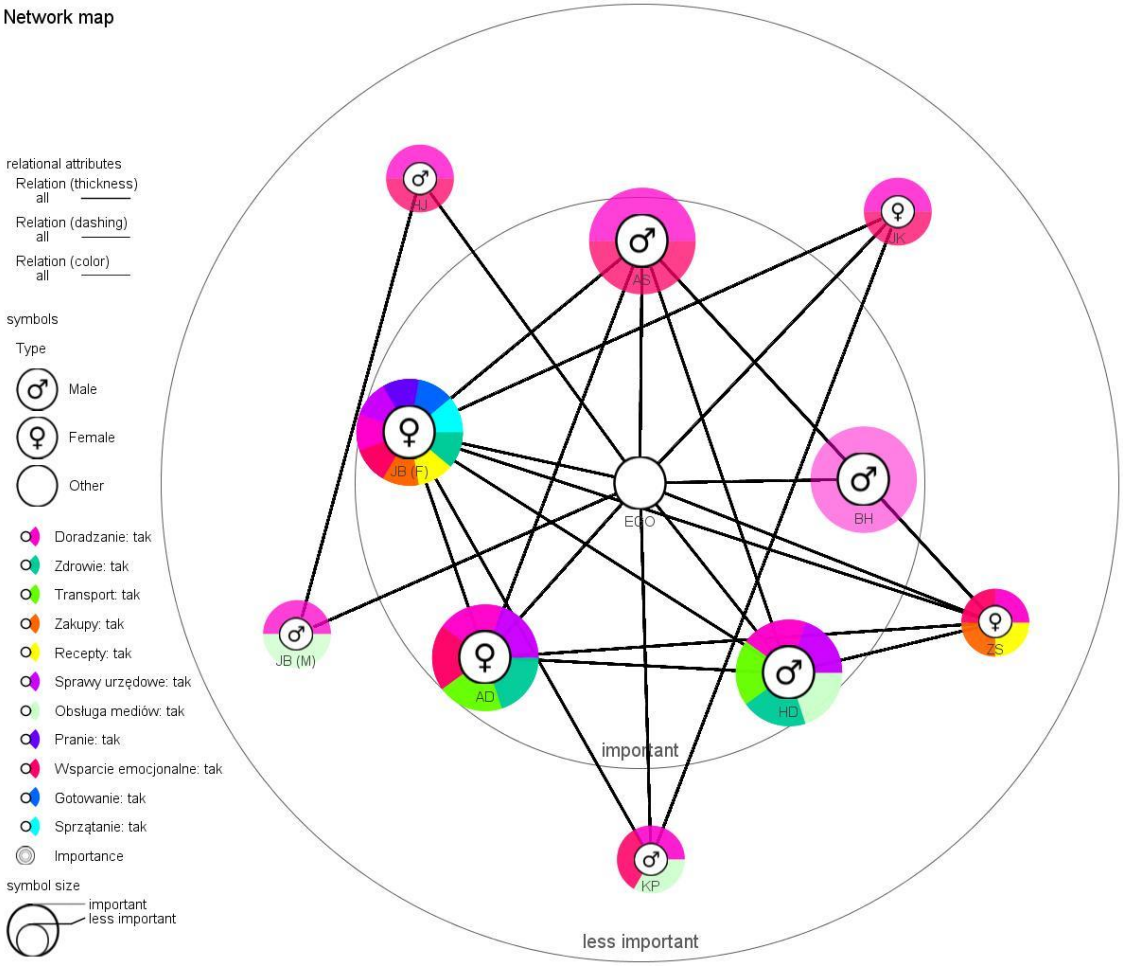
- **Zdrowie**

Problem	Życzenia/propozycje rozwiązania	Priorytet
Zapominanie o lekarstwach	System powiadomień	Wysoki

Sieci społeczne

Badania sieciowe obejmują analizę **ego** (badana osoba) oraz jej relacji z innymi aktorami (**alters**). W niniejszym badaniu użytkownicy końcowi mieli za zadanie określić, od kogo otrzymują wsparcie: w pierwszej kolejności badani wskazywali osoby, od których najczęściej otrzymują wsparcie (najważniejsze osoby) a następnie badani identyfikowali osoby, do których zwróciliby się o pomoc w sytuacji kryzysowej. Analiza sieci społecznych została wykonana w programie Vennmaker. Poniżej znajdują się szczegółowe wyniki badania pokazujące obszary wsparcia osób starszych testujących aplikację pomost.

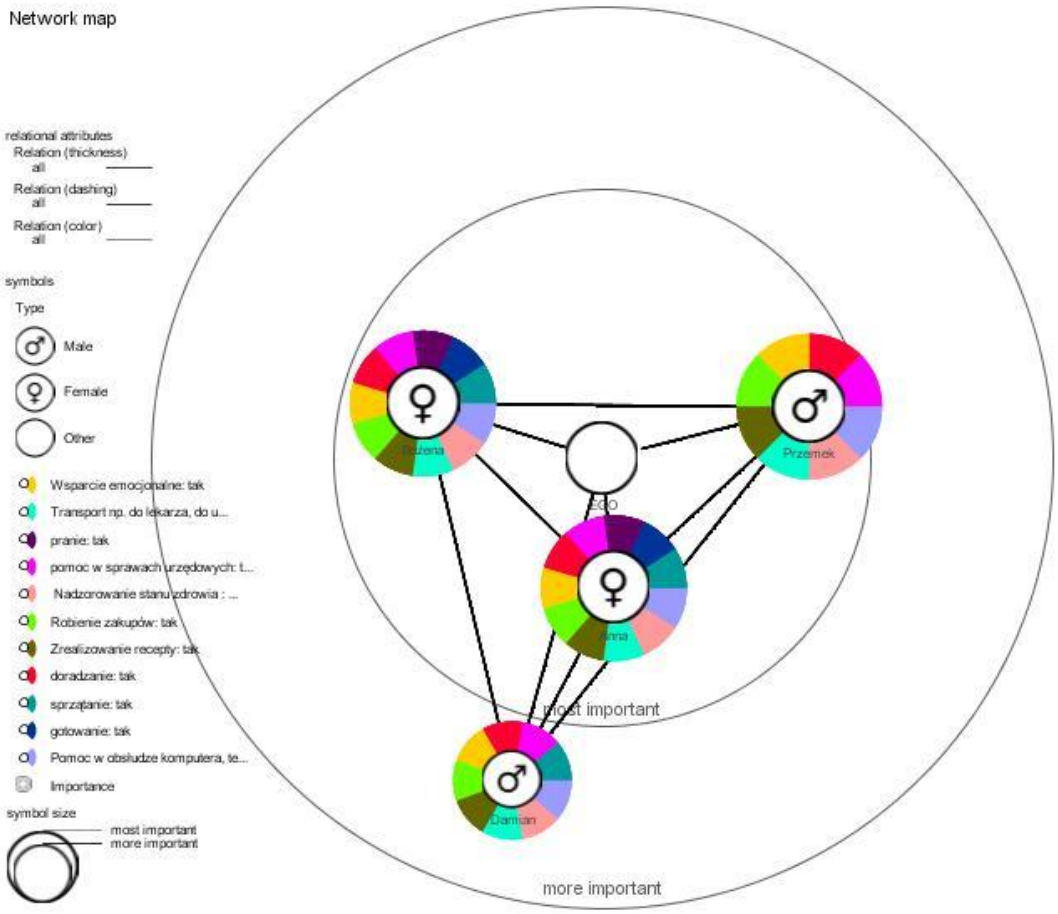
Przypadek nr 1. Mężczyzna, 72 lata, wykształcenie średnie



W kręgu osób, na których użytkownicy końcowi mogą najbardziej polegać znajdują się głównie członkowie rodziny. W szczególności partnerka badanego (JB) jest kluczową osobą w kwestii pomocy realizując w stosunku do męża praktycznie wszystkie formy wsparcia. Zaraz po niej badany wskazał najbliższą rodzina, czyli córkę i zięci (AD, HD). Z osób ważnych spoza rodziny badany wymienił

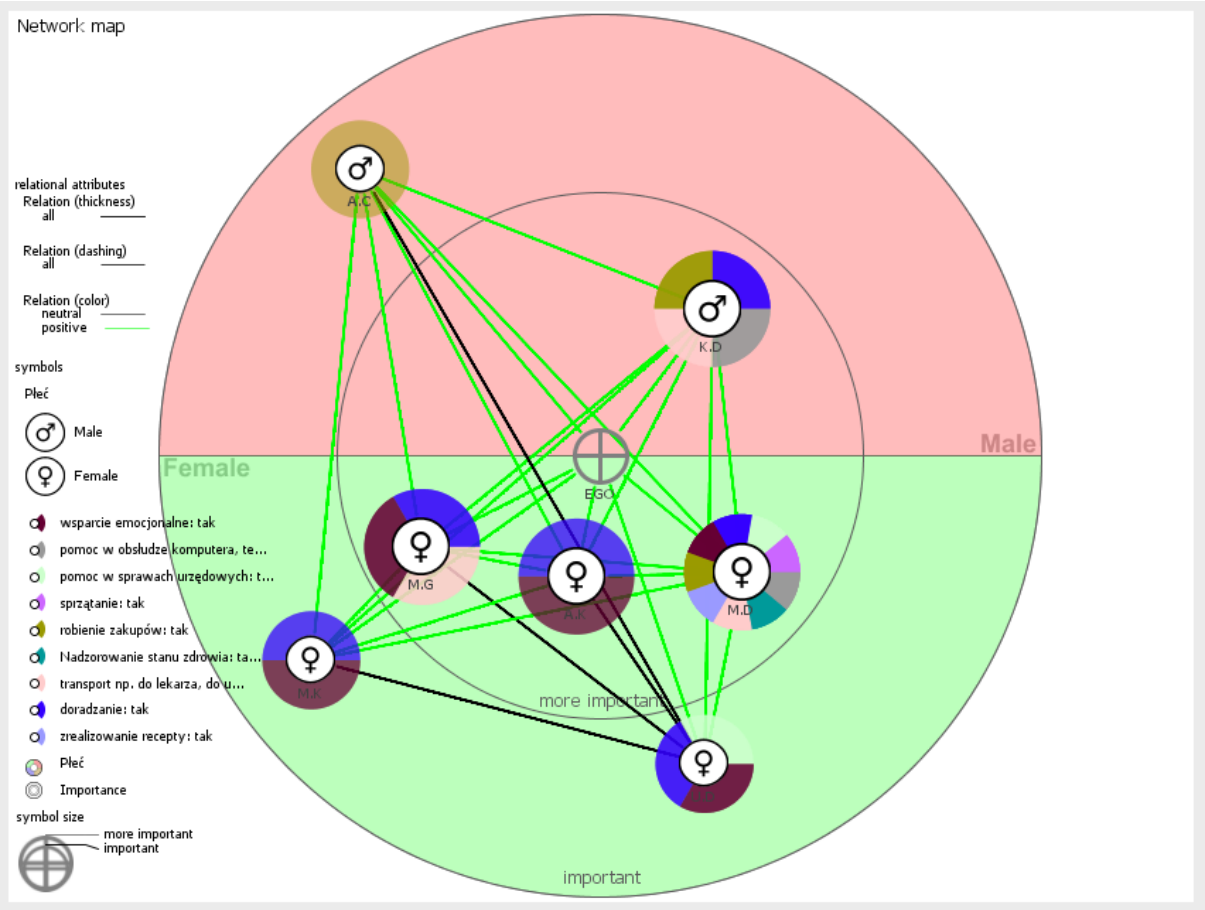
dwóch sąsiadów, od których otrzymuje pomoc głównie w dwóch kategoriach: doradzanie oraz wsparcie emocjonalne. Wymienione w pierwszym kręgu osoby cechuje duże zróżnicowanie w zakresie form udzielanej pomocy. Pozostałe osoby wymienione w drugim (mniej ważnym kręgu sieci wsparcia) charakteryzuje mniejsze zaangażowanie w pomoc badanemu. Wynika to, z faktu, iż są to osoby oddalone od miejsca zamieszkania Ego i kontakt odbywa się głównie poprzez telefon.

Przypadek nr 2. Kobieta, 87 lat, wykształcenie średnie



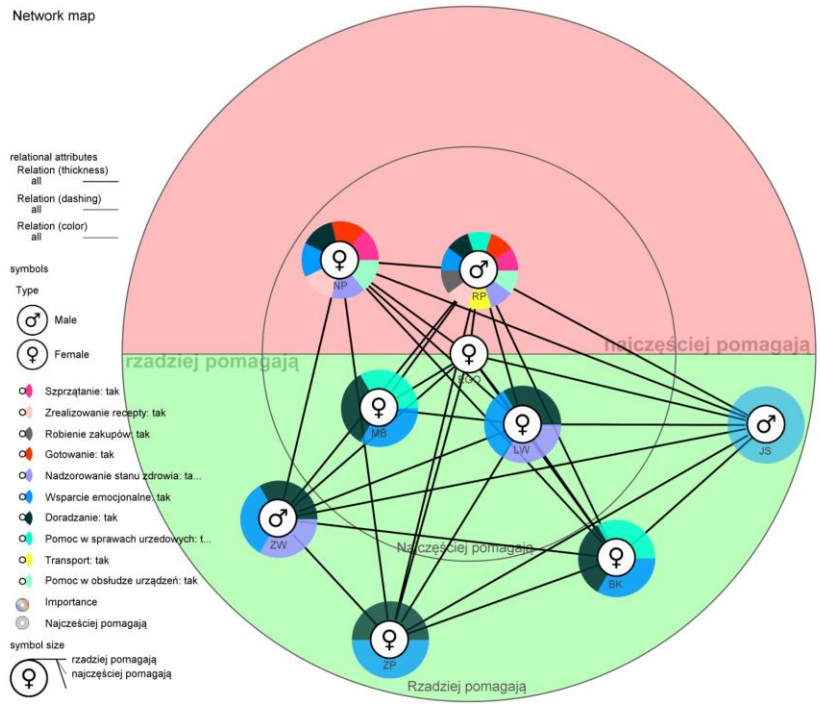
Respondentka ma 87 lat. Jej dzień wygląda zazwyczaj podobnie, w weekend spędza więcej czasu z wnukami. Potrzebuje stałej pomocy swoich bliskich podczas kąpieli lub spacerów, jednak stara się pomagać przy obowiązkach domowych oraz wspiera swoje dzieci w opiece nad wnukami. Nie utrzymuje bliskich kontaktów z sąsiadami, nie otrzymuje od nich pomocy, sama też im nie pomaga. Wynika to z tego, iż to najczęściej wnuki lub dzieci pomagają jej w potrzebie. Stąd wynika również jej niewielka sieć powiązań. Wnuki odpowiedzialne są głównie za sprawy związane z pomocą w obsłudze telefonu, Internetu. Z kolei dzieci pomagają respondentce praktycznie w wszystkich czynnościach związanych z codziennym funkcjonowaniem.

Przypadek nr 3. Kobieta, 78 lat, wykształcenie wyższe



W sieci tej badanej większość osób pomagających to kobiety, jednak tylko jedna kobieta ma więcej niż trzy obowiązki. Jest to córka osoby testującej aplikację i pomaga jej praktycznie we wszystkich czynnościach opiekuńczych. Najczęściej pojawiającym się obowiązkiem było doradzanie: pojawiło się ono u 6/7 członków sieci. Bardzo często pojawiał się również wsparcie emocjonalne oraz robienie zakupów. Syn badanej (AC) mieszka ok. 30km od respondentki i jego wsparcie polega na robieniu większych zakupów raz w tygodniu. Drugi syn mieszka w tym samym mieście i pomaga rodzicom w transporcie. Większość członków sieci ma pozytywne relacje między sobą. Jedynie kobiety oferują wsparcie emocjonalne, zarówno te z dalszego, jak i bliższego kręgu. Poza najbliższą rodziną badana wymienił swoich wnuków jako źródł wsparcia. Pomagają oni respondentce w obsłudze komputera oraz w sprawach urzędowych. W sieci badanej nie ma ani jednej osoby spoza rodziny.

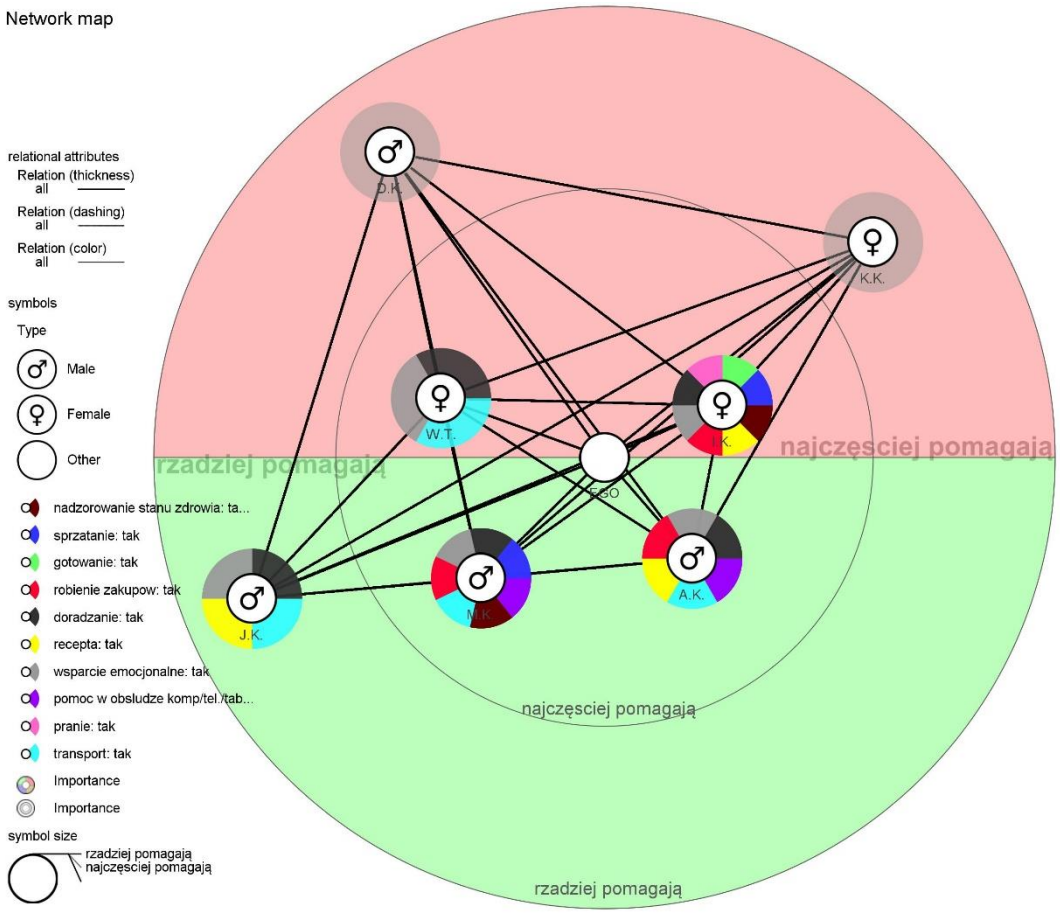
Przypadek nr 4. Kobieta, 73 lata, wykształcenie wyższe



Wielkość sieci wsparcia to 8 osób. Wśród wymienionych altersów przeważają osoby należące do rodziny i są to głownie kobiety. Tylko z trzema altersami łączą ego koleżeńskie stosunki. Zatem możemy wywnioskować, iż altersi to osoby, z którymi ego ma najwięcej wspólnych doświadczeń. Co więcej, sieć ta będzie wpływać na poczucie bezpieczeństwa ego w razie potrzeby pomocy. Kolejno relacje pomiędzy ego a altersami są w miarę stabilne w czasie, gdyż sieć składa się z osób, które ego zna bardzo długo. Najkrócej ego zna osobę 19 lat, a najdłużej 66 lat. Biorąc pod uwagę miejsce zamieszkania altersów, to prawie wszyscy altersi mieszkają blisko ego (w odległości do 5km). Tylko dwóch altersów mieszka dalej (jeden w odległości od 6 do 20km, a drugi powyżej 20km). Zatem ego najczęściej kontaktuje się z altersami osobiście, ale występuje także rozmowa telefoniczna, zwłaszcza gdy ktoś mieszka daleko. Ponadto jest to silna sieć (występuje częsty kontakt ego z altersami).

Pomimo iż ego cieszy się całkowitą sprawnością fizyczną, to analiza pokazuje, iż prawie w każdej czynności ego otrzymuje od kogoś pomoc. Jak się okazuje, żaden z altersów nie pomaga ego w robieniu prania. Najwięcej pomocy ego otrzymuje od dwóch altersów RP i NP (męża i bratanicy). Reszta altersów rzadko pomaga ego. Z racji tego, iż pośród wszystkich altersów tylko RP i NP pomagają ego w takich kwestiach, jak: gotowanie, sprzątanie, realizowanie recepty oraz obsługa komputera, to możemy przyjąć, iż wpływa na to bliskość miejsca zamieszkania ego. Zaś pozostali altersi w przewadze okazują wsparcie emocjonalne, a także pomagają w doradzaniu. Co ciekawe, tylko jeden alter (RP) pomaga w robieniu zakupów i transporcie do lekarza. Zaś alter JS oferuje tylko wsparcie emocjonalne.

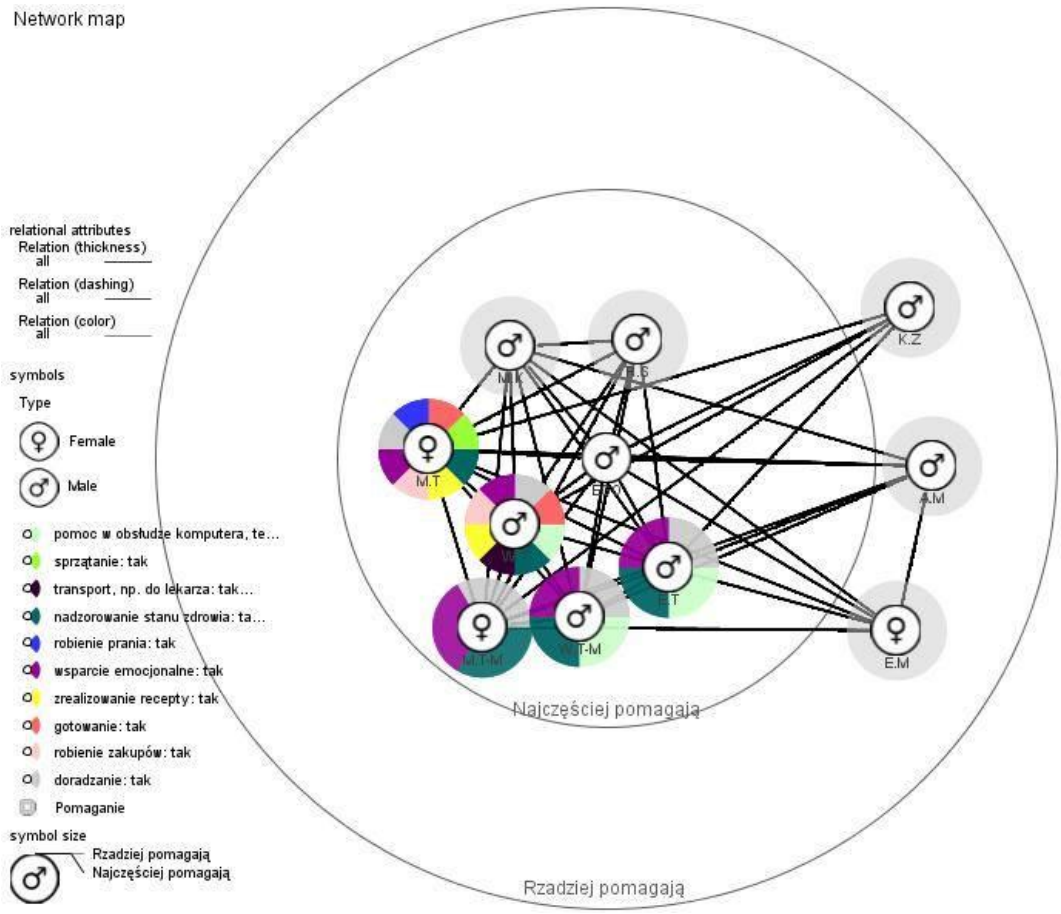
Przypadek nr 5. Mężczyzna, 75 lat, wykształcenie średnie.



W sieci ego znajduje się w większości najbliższa rodzina, wyjątkiem jest jedna sąsiadka. Osoby, które najbardziej pomagają, to te, które mieszkają z ego w domu: a więc jego żona oraz córka z rodziną. Ci, którzy mieszkają dalej, udzielają jedynie wsparcia emocjonalnego.

Pięć osób w sieci korzysta z Internetu. Ego wskazało, iż osoby te – głównie wnuk - pomagają mu oswoić się z nowymi technologiami.

Przypadek 6. Mężczyzna, 77 lat, wykształcenie wyższe.



Na podstawie schematu przedstawiającego sieć społeczną badanego ego, można stwierdzić, że sieć jest homogeniczna: ego jest mężczyzną i większość jego kontaktów również. Wyjątkiem jest między innymi małżonka (MT), która odpowiada z większość czynności opiekuńczych w stosunku do badanego. Gęstość wynosi niespełna 1, co oznacza, że osoby te w dużej mierze znają się nie tylko z ego, ale i z pozostałymi członkami sieci. Alters są osobami, które respondent wyminił jako te, do których zwraca się w razie potrzeby, od których może liczyć na pomoc i poza członkami rodziny obejmują byłych współpracowników oraz sąsiadów. W każdym przypadku kontakty badanego pomocne są w doradzaniu i wsparciu emocjonalnym. Sąsiedzi oraz byli współpracownicy pomagają badanemu głównie w zakresie doradzania.

Stosunek osób testujących do nowych technologii

Wszystkie osoby testujące aplikację mają bardzo pozytywny stosunek do nowych technologii komunikacyjnych. W zależności od stopnia korzystania z nowych technologii badani oceniają swoje umiejętności technologiczne jako dobre i bardzo dobre. Wykorzystanie nowych technologii w realizacji codziennych czynności zależy, zdaniem badanych, od spełnienia kilku warunków:

- Rozwiązania technologiczne powinny być proste w obsłudze i powinny działać szybko i z zapewnieniem bezpieczeństwa użytkownikom.
- Ważne dla badanych jest wsparcie. Większość badanych wskazywała na młodsze generacje jako źródło wsparcia. Proces uczenia się nowych technologii komunikacyjnych badani zazwyczaj określali jako trudny:

> ***"Nie boję się korzystania z Internetu, ale tylko wtedy, gdy mam wsparcie mojego wnuka".***

- Dla osób testujących aplikację bardzo ważna była cena nowych, bardziej zaawansowanych rozwiązań:

> ***"Pieniądze są problemem. Im bardziej zaawansowana technologia, tym wyższa cena. Nie stać mnie często na takie wydatki.***

- Strach przed oceną. Mimo że badani chętnie wprowadzają ulepszenie technologiczne do swojego życia, to obawiają się, że będą postrzegani przez otoczenie jako niesamodzielni, chorzy, niesprawni. Strach przed oceną wywołuje opór::

> ***"Nie wiem czy chciałabym zaawansowanych technologii w moim domu, nie chcę być postrzegana jako osoba, która nie radzi sobie z własnym życiem i musi się posiłkować „robotami".***

- Strachu przed nieznanym. Badani są otwarci na nowinki technologiczne – często dzięki młodszym członkom rodziny – jednak w pewnym stopniu obawiają się nieznanych konsekwencji oraz problemów związanych z bezpieczeństwem:

> ***"Czasami boję się kliknąć na coś, bo nie wiem, czego się spodziewać. Często komunikaty jakie się pojawiają są dla mnie niezrozumiałe".***

- Ochrona danych osobowych stanowi bardzo to zdaniem badanych bardzo ważny problem Wynika to w dużej mierze ostatnich doniesień medialnych dotyczących kupowania danych, nadużywania tych danych. Również dyskusja wokół GIODO sprawiła, że badani podchodzą do technologii i zbieranych przez nie informacji z bardzo dużą ostrożnością. Badani nie chcą, aby dane na ich temat, często dotyczące ich stanu zdrowia, były wykorzystywane przez nieznane im osoby:

> - ***"Ja nie chcę, żeby ktoś grzebal w informacjach o mnie. Pan by chciał żeby ktoś analizował Pańską kartę medyczną?"***

- Brak wiedzy na temat rozwiązań technologicznych dedykowanych osobom starszym. Badani wskazywali, że z pewnością są takie aplikacje, urządzenia, które by im pomogły w życiu, ale nie ma instytucji, która zajmowałaby się informowaniem o tym seniorów.

Profile użytkowników

Wnioski z badań pozwalają na przedstawienie trzech profile użytkowników, które różnią się w zakresie postawy wobec nowych technologii komunikacyjnych, znajomości nowych technologii oraz potrzeb. Profil jest modelem użytkownika aplikacji poMOST przedstawionym w opisowej formie umiejętności, potrzeb oraz z uwzględnieniem kontekstu społecznego. Innymi słowy, profil to archetyp użytkownika, który ma odzwierciedlać cechy charakterystyczne danej grupy użytkowników. Zadaniem konstruowanego profilu jest uspójnienie wiedzy nt. odbiorców projektu. Profil jest najczęściej używanym, standardowym narzędziem przy projektowaniu aplikacji i pozwala "wejść w buty użytkownika", a co za tym idzie, skupić się bardzo mocno na potrzebach, problemach oraz oczekiwaniach już nie odbiorców jako zbiorowości, ale jej pojedynczych przedstawicieli.

W procesie projektowania user experience przyjmuje się, że użytkownicy nie posiadają tylko jednego profilu, a często jest ich kilka do kilkunastu, w zależności od złożoności grupy docelowej.

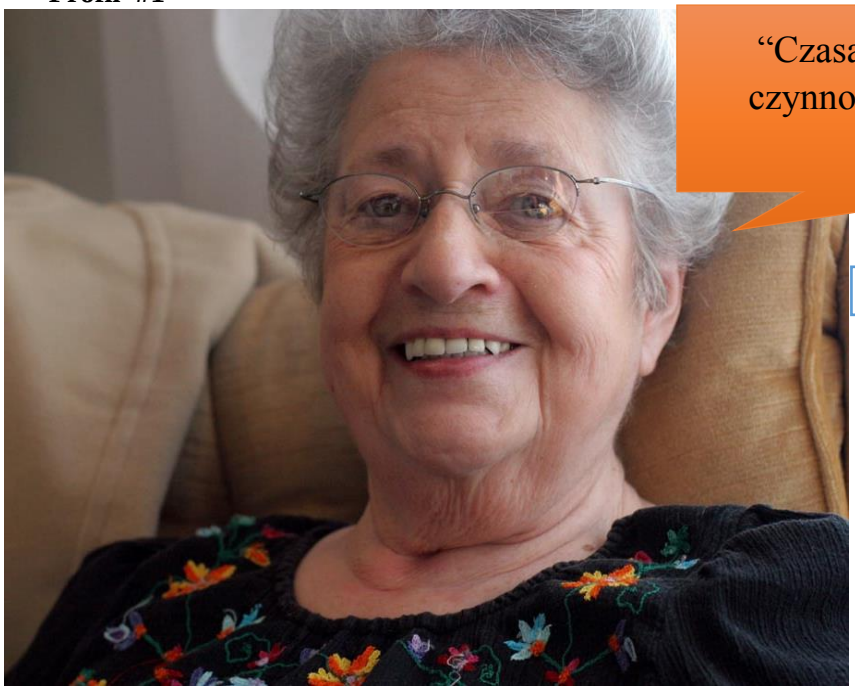
Profile składają się z charakterystyk zawierających:

- imię, wiek, status;
- krótki opis, biografia, rodzina, hobby, marzenia, cele życiowe;
- motto, charakterystyczne powiedzenie, cytat prezentujący postawę życiową;
- zdjęcie najlepiej odwzorowujące naszą personę (nigdy prawdziwe zdjęcie osoby badanej);
- umiejętności techniczne: staż korzystania z internetu, częstotliwość korzystania
- i wiele innych zależnych od projektu informacji.

Z względu na specyfikę aplikacji, profile użytkowników zostały uzupełnione m.in. o:

- oczekiwania i problemy grupy
- postawy wobec ICT
- tym wymaganej pomocy

Profil #1



"Czasami potrzebuję pomocy w drobnych czynnościach, ale nie chcę zawracać głowy dzieciom"

UMIEJETNOŚCI TECHNOLOGICZNE

KORZYSTA Z

MARIA NOWAK

WIEK: 73

STAN CYWILNY: ZAMĘŻNA

RODZINA: 2 DZIECI I 3 WNUKÓW

ZAWÓD: EMERYTOWANA NAUCZYCIELKA

POTRZEBUJE POMOCY: DROBNE ZAKUPY, WYPROWADZENIE PSA

AKTYWNOŚCI

O Marii

Maria jest emerytowaną nauczycielką geografii. Przed przejściem na emeryturę musiała nauczyć się obsługi komputera. Mimo iż potrafi posługiwać się smartfonem, to czyni to niechętnie i potrzebuje w tym zakresie pomocy. Mieszkanie Marii znajduje się na drugim piętrze w bloku bez windy. Ze względu na problem z biodrem rzadko opuszcza mieszkanie – 2-3 razy w tygodniu wychodzi ze swoim psem. Czas wolny Maria najchętniej spędza w mieszkaniu oglądając swoje ulubione seriale.
Jej znajomi, to przede wszystkim koleżanki z pracy, ale Maria rzadko się z nimi widuje ze względu na ograniczenia ruchowe.

STOSUNEK DO TECHNOLOGII

- Maria ma komputer oraz smartfon, ale wykorzystuje tylko podstawowe funkcje. Wiedze na temat ICT ma głównie dzięki wnukom.
- Boi się korzystać z Internetu sama; słyszała, że można zostać oszukanym i stracić pieniądze.

NIE LUBI

- Wysokich cen
- Zmywarki (trzeba się schylać)
- Gubienia drobnych przedmiotów
- Być sama

Fundusze Europejskie Wiedza Edukacja Rozwój
MAŁOPOLSKA
ROPS KRAKÓW INSTYTUCJA WOJEWÓDZTWA MAŁOPOLSKIEGO
Unia Europejska Europejski Fundusz Społeczny

Profil #2

"Wszystko jest dla ludzi. Internet jest bardzo pomocny, ale uczenie się nowych rzeczy jest czasami zbyt skomplikowane"

UMIEJETNOŚCI TECHNOLOGICZNE

KORZYSTA Z

ANDRZEJ KOWALSKI

WIEK: 77

STAN CYWILNY: WDOWIEC

RODZINA: 2 DZIECI

ZAWÓD: EMERYTOWANY PROFESOR

POTRZEBUJE POMOCY: WSKAZÓWKI JAK KORZYSTAĆ Z ICT, PARTNERÓW DO GRY W SZACHY, MYCIE OKIEN I SPRZĄTANIE

O ANDRZEJU

Andrzej korzysta z komputera od wielu lat. Jeszcze jako aktywny profesor nauczył się korzystania z Internetu i wyszukiwania informacji. Niemniej boi się nowinek technologicznych. W wolnym czasie lubi gotować oraz grać w szachy. Niestety większość jego znajomych nie żyje lub zmienili miejsce zamieszkania. W ostatnim czasie zaczął czytać e-booki. Dzieci kupują mu często prezenty związane z nowymi technologiami i uczą Andrzeja jak z nich korzystać. Zakupy robi online z dostawą do domu, ale problemem są drobne zakupy.

AKTYWNOŚCI

STOSUNEK DO TECHNOLOGII

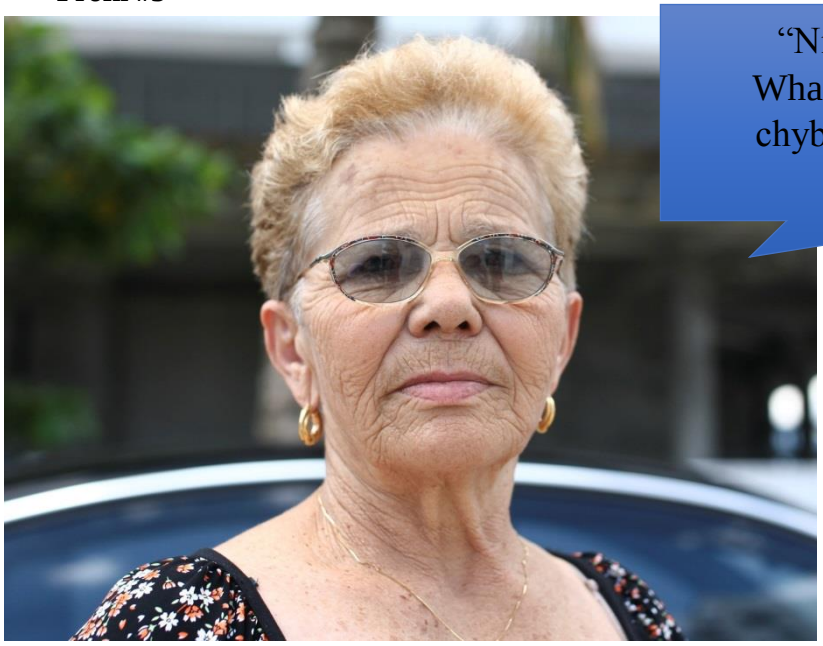
- Andrzej korzysta z komputera od wielu lat,
- W ostatnim czasie założył konto na Facebooku
- Czasami musi zadzwonić do swoich dzieci, aby mu pomogły w sprawach związanych z obsługą komputera i smartfona.

NIE LUBI

- Chodzić do lekarza
- Oglądać TV
- Czuć się zależnym od innych
- Sprzątania (szczególnie mycia okien)

Fundusze Europejskie Wiedza Edukacja Rozwój
MAŁOPOLSKA
ROPS KRAKÓW Regionalny Ośrodek Polityki Społecznej Instytucja Województwa Małopolskiego
Unia Europejska Europejski Fundusz Społeczny

Profil #3



"Nie mogę wyobrazić sobie życia bez Whatsappa, Facebooka i Google. Jestem chyba uzależniona od robienia zakupów online"

UMIEJETNOŚCI TECHNOLOGICZNE

KORZYSTA Z

TERESA MAŁECKA

WIEK: 75

STAN CYWILNY: WDOWA

RODZINA: 1 DZIECKO, 1 WNUCZKA

ZAWÓD: EMERYTOWANA KSIĘGOWA

POTRZEBUJE POMOCY: TRANSPORT, POCZTA.

O TERESIE

Teresa przez wiele lat pracowała jako księgowa w firmie męża. Dzięki temu doświadczeniu świetnie porusza się po świecie nowych technologii. Jest w stałym kontakcie z koleżankami ze szkoły – pomaga jej w tym Skype. Mimo iż nie ma większych problemów ze zdrowiem, to często traci orientację w terenie i boi się wyjść sama z domu. Nie może też długo stać w kolejce. W związku z tym często potrzebuje pomocy w zakresie transportu do lekarza czy też w odebraniu przesyłki z poczty. Teresa posiada bogatą kolekcję storczyków, którymi uwielbia się zajmować. Dzięki córce odkryła również zalety robienie zakupów online.

STOSUNEK DO TECHNOLOGII

- Nauczyła się obsługi komputera w pracy
- Na urodziny otrzymała od córki iPhone,
- Uwielbia robić zakupy online.

NIE LUBI

- kolejek
- dźwigania zakupów
- wizyt u lekarza

AKTYWNOŚCI

Fundusze Europejskie Wiedza Edukacja Rozwój
MAŁOPOLSKA
ROPS KRAKÓW INSTYTUCJA WOJEWÓDZTWA MAŁOPOLSKIEGO
Unia Europejska Europejski Fundusz Społeczny

Prezentacja wyników. Test użyteczności aplikacji

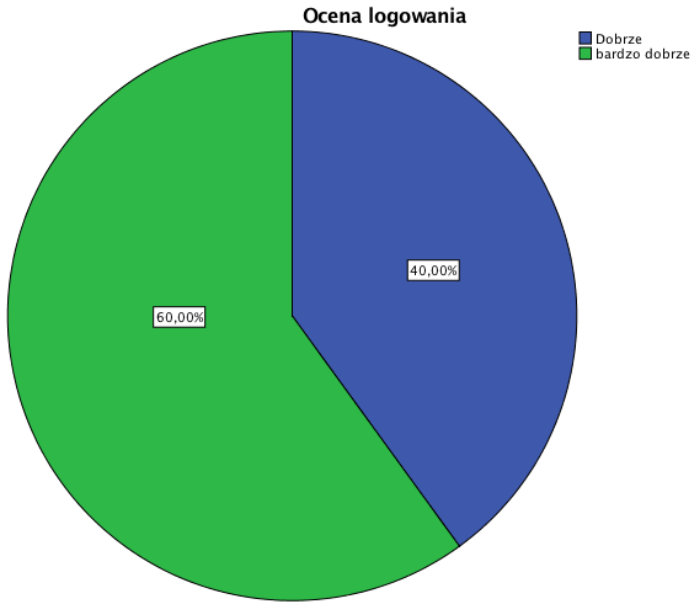
Ocena aplikacji poMOST przez osoby testujące oraz podjęte działania modyfikujące

Zgodnie ze specyfikacją innowacji 10 osób testowało aplikację poMOST. Wyniki analiz pokazują, że zdecydowana większość osób testujących bardzo dobrze ocenia główne funkcjonalności, wygląd oraz proces rejestracji i logowania.

Rejestracja i logowanie do aplikacji poMOST

Wszystkim osobom testującym udało się zarejestrować oraz logować do aplikacji. Sam proces rejestracji oceniany było jako bardzo prosty, ponieważ wymagane są minimalne i standardowe informacje. Badani stwierdzili, że zarówno rejestracja jak i logowanie są bardzo intuicyjne i nie ma konieczności korzystania z pomocy w tych czynnościach.

Wykres 1. Ocena procesu logowania do aplikacji poMOST



Czytelność komunikatów u wszystkich badanych została oceniona jako bardzo dobra.

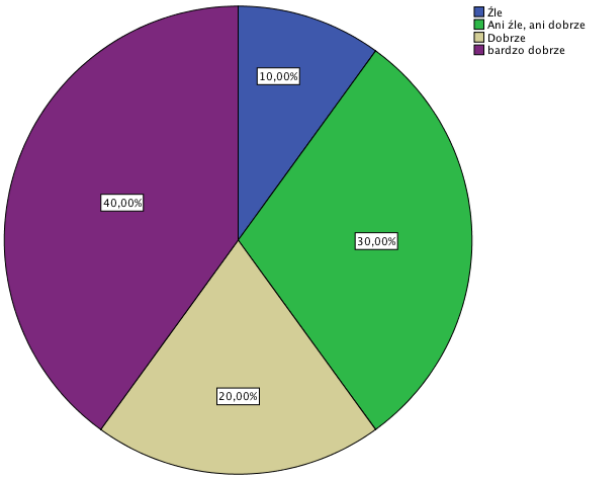
<u>Podjęte działania modyfikujące proces rejestracji i logowania</u>:

- Brak konieczności dokonywania zmian. System rejestracji i logowania został oceniony przez badanych bardzo wysoko.

Wygląd aplikacji

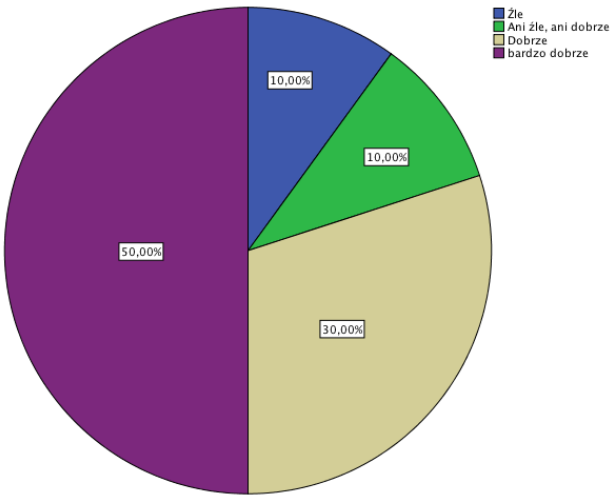
Ogólny wygląd aplikacji oceniony został przez badanych bardzo dobrze.

Wykres 2. Ocena kolorystyki aplikacji poMOST



Większość (60%) badanych ocenia kolorystykę aplikacji bardzo dobrze (40%) i dobrze (20%). Stosunkowo często pojawiały się w czasie testu stwierdzenia sprzeczne. Niektórzy badani wskazywali, że należy zwiększyć kontrast między kolorami, inni badani wskazywali, że kontrast jest za duży („*za duży kontrast między kolorami, sam niebieski byłby lepszy*"). Niektóre osoby testujące wyrażały konieczność całkowitej zmiany kolorów („*Lepszy byłby pomarańczowy*").

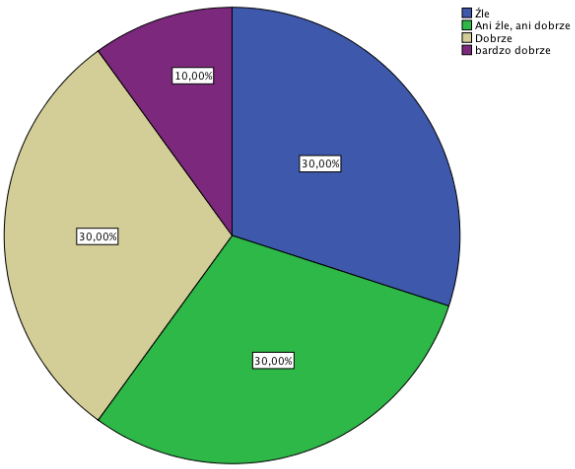
Wykres 3. Ocena czcionki zastosowanej w aplikacji



Powyższa tabela (Tab. 3) pokazuje, że czcionka zastosowana w aplikacji jest oceniana bardzo dobrze i dobrze.

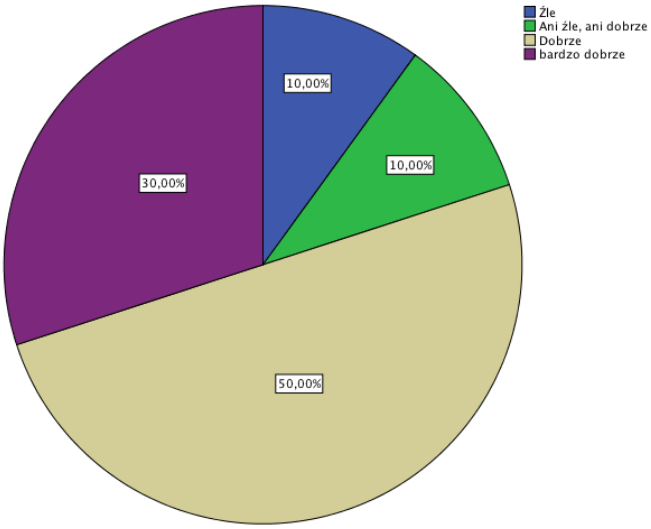
Oceniając wielkość czcionki, osoby testujące często wskazywały, że jest ona za mała:

Wykres 4. Ocena wielkości czcionki



Natomiast wykorzystane w aplikacji ikony i symbole (tab. 5) są dla badanych osób intuicyjne i nie ma konieczności dodawania instrukcji.

Wykres 5. Ocena ikon/symboli zastosowanych w aplikacji



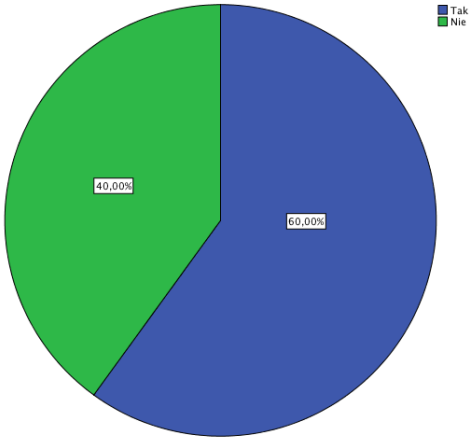
<u>Podjęte działania modyfikujące wygląd aplikacji</u>:

- Postanowiono nieco rozjaśnić kolor niebieski, ale z zachowaniem dotychczasowej kolorystyki.
- Wielkość czcionki została powiększona.

Funkcjonalność # 1: Zakupy

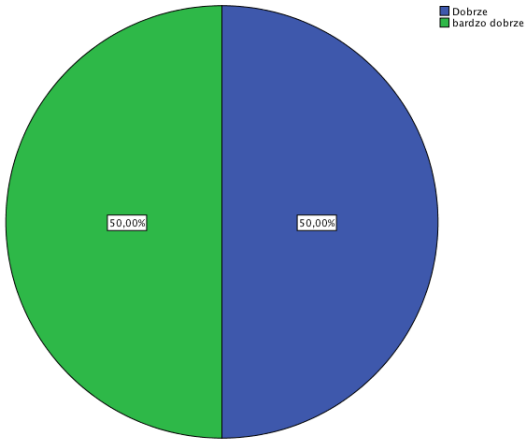
Większość osób testujących aplikację skorzystała z funkcji zakupów. Jest to funkcjonalność, z której badani najchętniej korzystali.

Wykres 6. Korzystanie z funkcji „Zakupy" przez osoby testujące



Sama funkcjonalność i jej przydatność zostały przez badanych, którzy skorzystali z tej opcji, oceniona bardzo dobrze i dobrze:

Wykres 7. Ocena funkcjonalności „Zakupy"



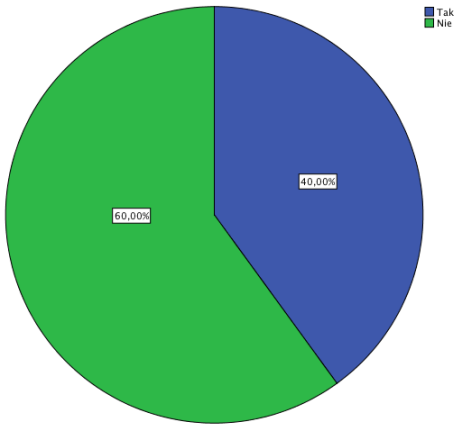
<u>Podjęte działania modyfikujące funkcjonalność „Zakupy"</u>:

- Brak konieczności dokonywania zmian. Badani bardzo dobrze oceniają tę funkcjonalność i nie zgłoszono uwag w stosunku do niej.

Funkcjonalność # 2: Transport

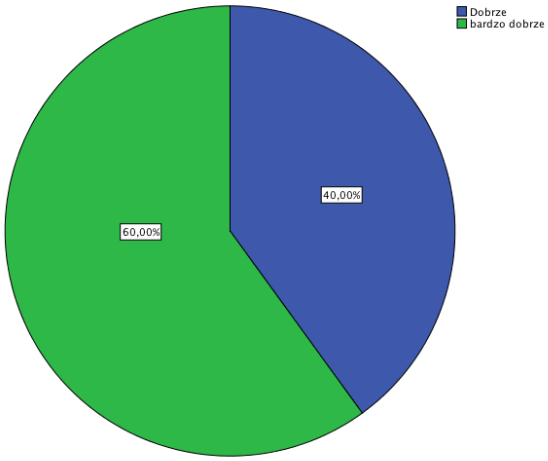
Drugą funkcjonalnością, z której osoby testujące aplikację korzystały był transport.

Wykres 8. Korzystanie z funkcji „Transport” przez osoby testujące



Badani skorzystali z tej funkcjonalności, aby dostać się do kościoła, lekarza, spotkania ze znajomymi oraz do supermarketu.

Wykres 9. Ocena funkcjonalności „Transport”



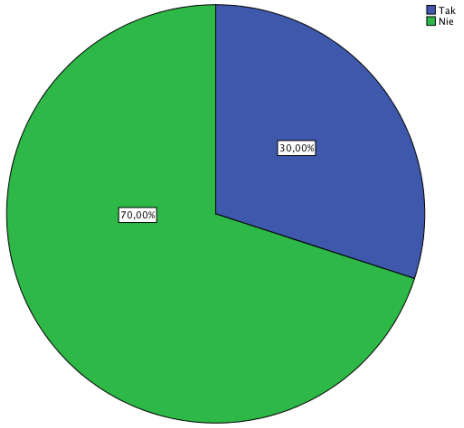
Podjęte działania modyfikujące funkcjonalność „Transport”:

- Brak konieczności dokonywania zmian. Badani bardzo dobrze oceniają tę funkcjonalność i nie zgłoszono uwag w stosunku do niej.

Funkcjonalność # 3: Internet i komputer

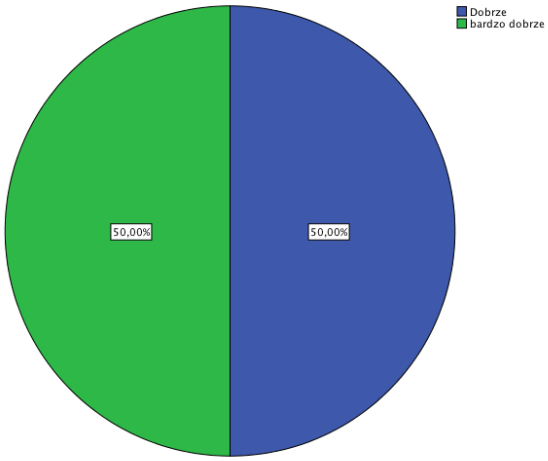
Trzecią najczęściej wybieraną funkcjonalności przez osoby testujące była opcja „Internet i komputer". W tym wypadku badani prosili o pomoc w zakresie problemów z połączeniem internetowym i instalacją programów.

Wykres 10. Korzystanie z funkcji „Internet i komputer" przez osoby testujące



Badani bardzo dobrze i dobrze ocenili tę funkcjonalność:

Wykres 11. Ocena przydatności funkcji „Internet i komputer"



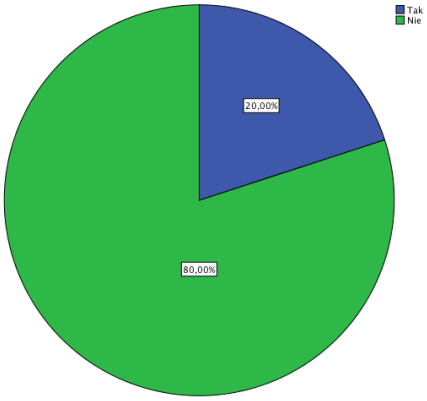
Podjęte działania modyfikujące funkcjonalność „Internet i komputer":

- Brak konieczności dokonywania zmian. Badani bardzo dobrze oceniają tę funkcjonalność i nie zgłoszono uwag w stosunku do niej.

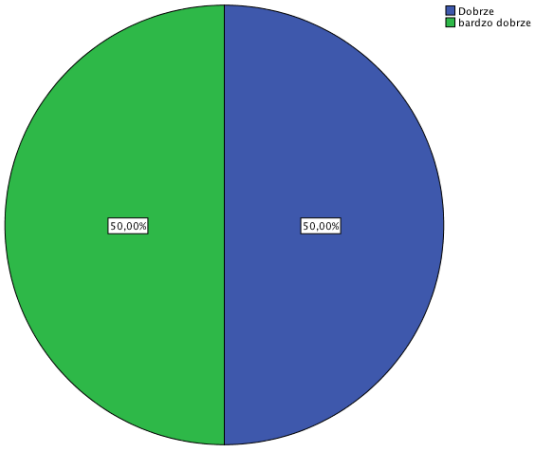
Funkcjonalność # 4: Poczta

Badani w niewielkim stopniu korzystali z opcji „Poczta”. Wynika to z faktu iż często, na przykład przy odbiorze przesyłki, konieczna jest obecność adresata.

Wykres 12. Korzystanie z funkcji „Poczta” przez osoby testujące



Wykres 13. Ocena funkcjonalności „Poczta”



Ci badani, którzy skorzystali z funkcji „Poczta” oceniają ją bardzo dobrze i dobrze.

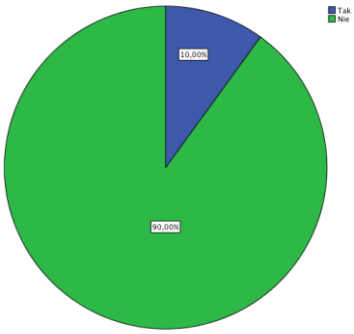
<u>Podjęte działania modyfikujące funkcjonalność „Poczta”</u>:

- Brak konieczności dokonywania zmian. Badani bardzo dobrze oceniają tę funkcjonalność i nie zgłoszono uwag w stosunku do niej.

Funkcjonalność # 5: Zdrowie

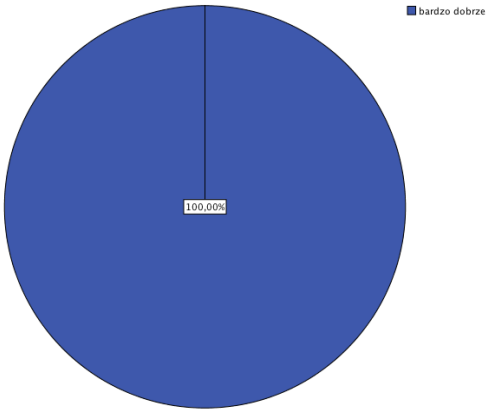
Osoby testujące aplikację poMOST bardzo rzadko korzystały z funkcjonalności „Zdrowie”:

Wykres 14. Korzystanie z funkcji „Zdrowie” przez osoby testujące



Niewielkie zainteresowanie tą funkcjonalnością wynika z tego, że zdrowie jest postrzegane przez badanych jako coś osobistego. Niemniej te osoby testujące, które skorzystały z opcji „Zdrowie” oceniają ją bardzo dobrze:

Wykres 15. Ocena funkcjonalności „Zdrowie”



<u>Podjęte działania modyfikujące funkcjonalność „Zdrowie”</u>:

- Brak konieczności dokonywania zmian. Badani bardzo dobrze oceniają tę funkcjonalność i nie zgłoszono uwag w stosunku do niej.

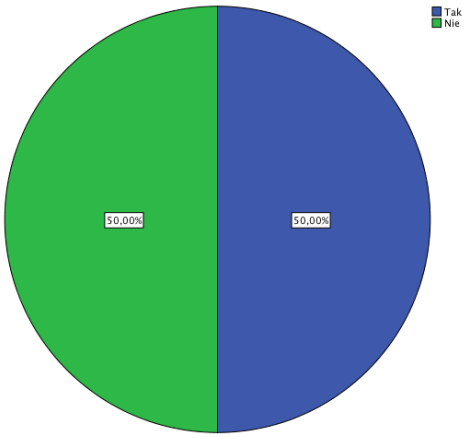
Funkcjonalność # 6: Alarm

Najrzadziej wybieraną funkcjonalnością był „Alarm”. Opcja ta została wykorzystana jedynie raz w charakterze testowym (bez realnego zagrożenia życia). Z jednej strony badani podkreślali, że jest to zbędna funkcja, ponieważ mają oni zazwyczaj zapisaną osobę pod numerem alarmowym i w sytuacji kryzysowej skorzystaliby z tego, a nie z aplikacji („*Alarm nie jest potrzebny, bo mam zapisany numer do wnuków, którym ufam*”). Z drugiej jednak strony część badanych uważa, że jest to potrzebna opcja i należy ją zostawić.

Zmiana ustawień

Ważnym aspektem przyjaznej aplikacji jest łatwy i przejrzysty proces zmiany ustawień. Jest to szczególnie ważne w przypadku seniorów.

Wykres 16. Zmiana ustawień w okresie testowania aplikacji:



Badani zwracali uwagę na jeden problem: trudność w zakresie zmiany roli osoby pomagającej na osobę potrzebującą pomocy.

<u>Podjęte działania modyfikujące ustawienia aplikacji</u>:

- Dodano nową rolę „Potrzebuję pomocy i chętnie pomogę” dzięki czemu użytkownicy, którzy potrzebują pomocy w pewnych czynnościach, ale mogą pomóc w innych, mogą korzystać w pełni z aplikacji.

Rekomendacje i zalecenia, w tym dla opiekunów osób zależnych

1. Aplikacja poMOST służy wspieraniu aktywizacji i niezależności osób starszych poprzez tworzenie lokalnych sieci społecznych, zrzeszających użytkowników – seniorów potrzebujących pomocy i użytkowników – wolontariuszy, którzy tej pomocy mogą udzielić. W procesie rejestracji istnieje możliwość wybrania podwójnej roli: osoby pomagającej i potrzebującej pomocy równocześnie.

2. Beneficjentami pomocy sąsiedzkiej powinny być osoby, które przynajmniej w minimalnym stopniu mają kontakt z dowolnymi urządzeniami mobilnymi lub komputerem - cechuje ich szybsza adaptacja do nowych urządzeń. Niekoniecznie muszą to osoby zupełnie niemobilne, natomiast cechuje je problem z poruszaniem się na znaczne odległości. Jednocześnie są to osoby aktywne, biorące udział w życiu kulturalnym i społecznym - dlatego warto szukać grup docelowych wśród Uniwersytetów Trzeciego Wieku, szkół seniorów, kółek zainteresowań, zrzeszeń przy związkach wyznaniowych.

3. Aplikacja pozwala na wymianę informacji pomiędzy użytkownikami poMOST; ułatwia tworzenie lokalnych sieci społecznych zrzeszających użytkowników seniorów i użytkowników wolontariuszy; ułatwia użytkownikom-seniorom formułowania zapytań o pomoc w drobnych sprawach życia codziennego, w szczególności w zakresie zrobienia zakupów, zapisania do lekarza, transportu, poczty, Internetu i komputera, alarmu.

4. Funkcja „Alarm" nie zastępuje w żadnym zakresie numerów alarmowych, w szczególności nr alarmowego 112. Jej prawidłowe funkcjonowanie jest w całości zależne od obecności w najbliższej okolicy użytkownika innych użytkowników oraz ich aktywności.

5. Aplikacja poMOST działa na systemie Android i tylko na urządzeniach mobilnych typu smartfon. Do uruchomienia i prawidłowego działania poMOST niezbędne jest aktywowanie następujących funkcji urządzenia mobilnego:

    a. aktywne połączenie internetowe,

    b. aktywną usługę GPS.

6. Przy pierwszym uruchomieniu poMOST użytkownik musi dokonać rejestracji poprzez podanie następujących danych: unikalna nazwa użytkownika (login), hasła, nr telefonu, adresu e – mail. Podanie danych osobowych jest dobrowolne, jednak odmowa ich podania uniemożliwia rejestrację, a co za tym idzie skorzystanie z poMOST.

7. W celu efektywnego korzystania z aplikacji konieczna jest aktywizacja sieci pozarodzinnej: sąsiedzkiej/koleżeńskiej, której członkowie będą mogli odpowiedzieć na problemy osoby starszej. Skutecznym sposobem jest poinformowanie sąsiadów – na przykład za pośrednictwem wspólnoty mieszkaniowej lub spółdzielni - o aplikacji i osobie, która może za jej pośrednictwem poprosić o drobną pomoc.

8. Ważnym aspektem zwiększającym zaangażowane wolontariuszy jest zwrócenie uwagi na bezpłatny charakter aplikacji oraz charakter pomocy, która obejmuje realizację drobnych „dobrych uczynków" przy okazji np. wracając z pracy, robiąc zakupy dla siebie.

9. W celu zapewnienia bezpieczeństwa użytkowników – w szczególności osób potrzebujących pomocy - na etapie rejestracji rekomenduje się wgranie zdjęcia tak, aby osoba starsza mogła

rozpoznać swoich sąsiadów i tym samym zminimalizować ryzyko pojawienia się zachowań zagrażających zdrowiu i życiu osoby potrzebującej pomocy.

10. Skuteczność aplikacji uwarunkowana jest powołaniem i działaniem instytucji zajmującej się zarządzaniem siecią wsparcia, rekrutacją wolontariuszy, dbaniem o bezpieczeństwo użytkowników, promowaniem aplikacji i pobudzaniem oddolnej pomocy sąsiedzkiej. Instytucja zarządzająca powinna odpowiadać za stworzenie bazy danych umieszczonej na serwerze. Sensowność powołania takiej instytucji wiąże się również z obowiązkiem ochrony danych osobowych gromadzonych w aplikacji. Instytucją organizującą sieć wsparcia w ramach aplikacji mogłaby być na przykład gmina, parafia, organizacja pozarządowa lub instytucja opiekuńcza.

11. Rekomendowany rozwiązaniem jest powołanie lokalnych przedstawicieli w ramach instytucji zarządzającej, osób, które będą w stanie pomóc – w szczególności starszym użytkownikom – w obsłudze aplikacji oraz bieżącym rozwiązywaniu problemów i potencjalnych konfliktów.

12. Instytucja zarządzająca aplikacją powinna również prowadzić szkolenia w zakresie bezpiecznego korzystania z aplikacji oraz samego jej funkcjonowania. Akcje promocyjne oraz edukacyjne powinny obejmować nie tylko potencjalnych wolontariuszy oraz osoby potrzebujące pomocy, ale również ich głównych opiekunów, członków rodziny.

13. Ze względu na fakt, iż skuteczność aplikacji zależy od liczby aktywnych użytkowników, instytucja zarządzająca powinna prowadzić stałą akcję promocyjną zachęcającą do udzielania wsparcia osobom starszym.

14. Wśród osób pomagających znaleźć się powinni również usługodawcy np. właściciele lokalnych sklepów, apteki, które mogłyby dostarczać produkty bezpośrednio do mieszkań osób starszych. Rozwiązanie takie wymagałoby jednak stworzenia dodatkowych regulacji określających odpłatność za świadczone usługi.

15. Sieć sąsiedzką można rozszerzyć o inne kręgi takie jak: młodzież szkolną, organizacje pozarządowe, byłych współpracowników, parafię. Niemniej pamiętać należy o procesie weryfikacji wolontariuszy przez instytucję, która zajmować się będzie zarządzaniem siecią wsparcia oraz dbaniem o bezpieczeństwo użytkowników i ochronę danych osobowych.

Podsumowanie

Test aplikacji został przeprowadzony w warunkach semi-naturalnych i kontrolowanych. Wynika to z faktu, iż w okresie testowym aplikacja nie mogła zostać upubliczniona i swobodnie wykorzystywana przez zainteresowanych użytkowników. Zarówno użytkownicy-osoby starsze, jak i użytkownicy-wolontariusze, to osoby, które się znają, co zniwelowało problemy związane z bezpieczeństwem użytkowników. Ze względu na fakt, iż aplikacja nie jest w tym momencie powiązana z żadną instytucją, która zapewniałaby jej bezpieczne wykorzystanie oraz zarządzałaby siecią wsparcia, niekontrolowane wykorzystanie aplikacji mogłoby prowadzić do licznych problemów natury prawnej.

Osoby testujące aplikację opierają swoją sieć wsparcia na rodzinie, której członkowie to główni opiekunowie. Niemniej badani często wskazywali, że w kwestii drobnych czynności opiekuńczych, czynności ułatwiających życie i wpływających na jego komfort często czują dyskomfort w proszeniu o pomoc kogoś z rodziny. Wynika to z faktu, iż obecnie członkowie rodziny są zazwyczaj rozproszeni geograficznie. Nawet jeśli w tym samym mieście mieszkają rodzinni opiekunowie, to czas dojazdu oraz obciążenie pracą i często własnym obowiązkami rodzicielskimi sprawiają, że proszenie o drobną przysługę jest dla osób starszych kłopotliwe. Siła aplikacji polega zatem na aktywizacji sieci pozarodzinnej i wykorzystania jej potencjału opiekuńczego dostępnego „na miejscu" i w bezpośredniej bliskości potencjalnych osób pomagających i potrzebujących pomocy.

Zrealizowane badania testowe pokazały, iż użytkownicy końcowi chętnie korzystają z aplikacji z względu na jej: dyskretny (częściowo anonimowy) charakter, prostą nawigację po systemie oraz brak kosztów. Aplikacja poMOST została oceniona przez grupę testującą bardzo wysoko. Zarówno predefiniowane obszary wsparcia, jak i sam interfejs zostały dostosowane do potrzeb osób starszych. Najczęściej wybieraną opcją były „Zakupy", „Transport" oraz „Internet i komputer". Ze względu na charakter funkcjonalności „Alarm" opcja ta nie została ani raz wybrana w sposób „naturalny" a jedynie w ramach testu sprawdzającego poprawność funkcjonowania.

Test aplikacji przeprowadzony w kontekście rutynowych działań codziennych wskazuje co prawda na liczne dodatkowe obszary, gdzie osoby starsze wymagają pomocy. Jednak problemy te wykraczają poza możliwości pomocowe aplikacji mobilnej, ponieważ bardzo często dotyczą architektury mieszkania. Tym samym lista predefiniowanych funkcjonalności nie uległa zmianie. Modyfikacji poddano jedynie wygląd aplikacji zwiększając tym samym jej czytelność oraz intuicyjność nawigacji po systemie.

Powodzenie wdrożeniowe aplikacji zależy, co zostało wielokrotnie zaznaczone w niniejszym raporcie, przede wszystkim od skuteczności instytucji, która będzie realizować obowiązki związane z: szeroką promocją aplikacji, rekrutacją wolontariuszy, weryfikacją wolontariuszy, szkoleniami osób starszych, rozwiązywaniem bieżących problemów technicznych, przeciwdziałaniem i ewentualnie rozwiązywaniem konfliktów, zarządzaniem bazami danych oraz ochroną danych osobowych. Tego typu podmiot mógłby działać na przykład w ramach gminy, ośrodków pomocy społecznej, organizacji pozarządowych, związków wyznaniowych.

Dokumentacja techniczna - kod

poMOST działa na systemie operacyjnym Android – najpopularniejszym oprogramowaniu stosowanym przez Polaków. Aplikacja została napisana w języku programowania Java. Język ten jest w tej chwili najbardziej popularnym językiem programowania. Pozwala w szybki i mało inwazyjny sposób dokonać wszelkich zmian i unowocześnień w aplikacji. Poniżej prezentuje kod aplikacji zachowując oryginalną pisownię oraz układ.

Appki

Droid:

MainActivity:

```
using System;

using Android.App;
using Android.Content;
using Android.Content.PM;
using Android.Runtime;
using Android.Views;
using Android.Widget;
using Android.OS;
using ImageCircle.Forms.Plugin.Droid;
using SVG.Forms.Plugin.Droid;
using FFImageLoading.Forms.Droid;
using FFImageLoading.Svg.Forms;
using Plugin.Permissions;
using XamEffects;
using Plugin.SecureStorage;

namespace PoMOST.Droid
{
```

```
[Activity(Label = "PoMOST", Icon = "@drawable/icon", Theme = "@style/MyTheme", MainLauncher = false, ConfigurationChanges = ConfigChanges.ScreenSize | ConfigChanges.Orientation,ScreenOrientation = ScreenOrientation.Portrait)]
public class MainActivity : global::Xamarin.Forms.Platform.Android.FormsAppCompatActivity
{
    protected override void OnCreate(Bundle bundle)
    {
        TabLayoutResource = Resource.Layout.Tabbar;
        ToolbarResource = Resource.Layout.Toolbar;

        base.OnCreate(bundle);

        global::Xamarin.Forms.Forms.Init(this, bundle);
        RoundedBoxView.Forms.Plugin.Droid.RoundedBoxViewRenderer.Init();

        ImageCircleRenderer.Init();
        CachedImageRenderer.Init();
        TouchEffect.Init();
        var ignore = typeof(SvgCachedImage);

        SecureStorageImplementation.StoragePassword = "POMOST";

        var width = Resources.DisplayMetrics.WidthPixels;
        var height = Resources.DisplayMetrics.HeightPixels;
        var density = Resources.DisplayMetrics.Density;

        App.ScreenWidth = (width - 0.5f) / density;
        App.ScreenHeight = (height - 0.5f) / density;

        LoadApplication(new App());
    }

    public override void OnRequestPermissionsResult(int requestCode, string[] permissions, [GeneratedEnum] Android.Content.PM.Permission[] grantResults)
    {
```

```
            base.OnRequestPermissionsResult(requestCode, permissions, grantResults);
            PermissionsImplementation.Current.OnRequestPermissionsResult(requestCode,
permissions, grantResults);
        }


        public event Action<int, Result, Intent> ActivityResult;

        protected override void OnActivityResult(int requestCode, Result resultCode, Intent data)
        {
            if (this.ActivityResult != null)
                this.ActivityResult(requestCode, resultCode, data);
        }
    }
}
```

MainApplication:

```
using System;

using Android.App;
using Android.OS;
using Android.Runtime;
using Plugin.CurrentActivity;
using Plugin.FirebasePushNotification;

namespace PoMOST.Droid
{
        //You can specify additional application information in this attribute
    [Application]
    public class MainApplication : Application, Application.IActivityLifecycleCallbacks
    {
        public MainApplication(IntPtr handle, JniHandleOwnership transer)
          :base(handle, transer)
```

```
    {
    }

    public override void OnCreate()
    {
        base.OnCreate();
        RegisterActivityLifecycleCallbacks(this);
        #if DEBUG
        FirebasePushNotificationManager.Initialize(this, true);
#else
        FirebasePushNotificationManager.Initialize(this,false);
#endif
    }

    public override void OnTerminate()
    {
        base.OnTerminate();
        UnregisterActivityLifecycleCallbacks(this);
    }

    public void OnActivityCreated(Activity activity, Bundle savedInstanceState)
    {
        CrossCurrentActivity.Current.Activity = activity;
    }

    public void OnActivityDestroyed(Activity activity)
    {
    }

    public void OnActivityPaused(Activity activity)
    {
    }

    public void OnActivityResumed(Activity activity)
```

```
        {
            CrossCurrentActivity.Current.Activity = activity;
        }

        public void OnActivitySaveInstanceState(Activity activity, Bundle outState)
        {
        }

        public void OnActivityStarted(Activity activity)
        {
            CrossCurrentActivity.Current.Activity = activity;
        }

        public void OnActivityStopped(Activity activity)
        {
        }
    }


}
```

Renderers:

FlatButtonRenderer:

```
using System;

using Android.Content;
using PoMOST.Droid.Renderers;
using Xamarin.Forms;
using Xamarin.Forms.Platform.Android;

[assembly: ExportRenderer(typeof(Button), typeof(FlatButtonRenderer))]
namespace PoMOST.Droid.Renderers
```

```
{
    public class FlatButtonRenderer : ButtonRenderer
    {
        public FlatButtonRenderer(Context context) : base(context)
        {
        }

        protected override void OnDraw(Android.Graphics.Canvas canvas)
                {
                        base.OnDraw(canvas);
                }

                protected override void OnElementChanged(ElementChangedEventArgs<Button> e)
                {
                        base.OnElementChanged(e);
                }
    }
}
```

GradientButtonRenderer:

```
using System;
using Android.Content;
using Android.Graphics.Drawables;
using Android.Support.V4.View;
using Android.Views;
using PoMOST.Droid;
using PoMOST.Views.Controls;
using Xamarin.Forms;
using Xamarin.Forms.Platform.Android;

[assembly: ExportRenderer(typeof(GradientButton), typeof(GradientButtonRenderer))]

namespace PoMOST.Droid
```

```
{
    public class GradientButtonRenderer : ButtonRenderer
    {
        private Xamarin.Forms.Color StartColor { get; set; }
        private Xamarin.Forms.Color EndColor { get; set; }
        private Xamarin.Forms.Color StartTouchColor { get; set; }
        private Xamarin.Forms.Color EndTouchColor { get; set; }

        Android.Widget.Button thisButton;

        GradientDrawable gradient;
        GradientDrawable gradientTouch;

        public GradientButtonRenderer(Context context) : base(context)
        {
        }



        public GradientDrawable DrawGradient(ElementChangedEventArgs<Xamarin.Forms.Button>
e)
        {
            var btn = e.NewElement as GradientButton;
            gradient = new GradientDrawable(GradientDrawable.Orientation.BlTr, new[] {
                        btn.StartColor.ToAndroid().ToArgb(),
                        btn.EndColor.ToAndroid().ToArgb()
                });


            //gradient.SetGradientRadius(0f);
            gradient.SetCornerRadius(20);
            gradient.SetStroke(0, btn.StartColor.ToAndroid());
```

```
        return gradient;
    }

    public GradientDrawable DrawGradientTouch(ElementChangedEventArgs<Xamarin.Forms.Button> e)
    {
        var btn = e.NewElement as GradientButton;
        gradientTouch = new GradientDrawable(GradientDrawable.Orientation.BlTr, new[] {
                    btn.StartTouchColor.ToAndroid().ToArgb(),
                    btn.EndTouchColor.ToAndroid().ToArgb()
            });
        gradientTouch.SetCornerRadius(20);
        gradientTouch.SetStroke(0, btn.StartColor.ToAndroid());

        return gradientTouch;
    }

    protected override void OnElementChanged(ElementChangedEventArgs<Xamarin.Forms.Button> e)
    {
        base.OnElementChanged(e);


       // thisButton.Click += HandleButtonClicked;

        if (e.OldElement != null || Element == null)
        {
            return;
        }
        try
        {
            thisButton = Control as Android.Widget.Button;
            //thisButton.SetBackgroundResource(Resource.Drawable.btn_unpress);
            thisButton.Touch += ThisButton_Touch;
```

```
            Control.StateListAnimator = new Android.Animation.StateListAnimator();

            Control.SetBackground(DrawGradient(e));

            gradientTouch = DrawGradientTouch(e);
        }
        catch (Exception ex)
        {
            System.Diagnostics.Debug.WriteLine(@"ERROR: ", ex.Message);
        }
    }

    private void ThisButton_Touch(object sender, TouchEventArgs e)
    {
        //System.Diagnostics.Debug.WriteLine("eeeeee");
        e.Handled = false;
        if (e.Event.Action == MotionEventActions.Down)
        {
            //System.Diagnostics.Debug.WriteLine("TouchDownEvent");
            if (gradient != null)
            {
                //gradient.Alpha = 50; //0-255 zakres
                thisButton.SetBackground(gradientTouch);
            }
            //thisButton.SetBackgroundColor(Android.Graphics.Color.Gray);
        }
        else if (e.Event.Action == MotionEventActions.Up)
        {
            //System.Diagnostics.Debug.WriteLine("TouchUpEvent");
            if (gradientTouch != null)
            {
                //gradientTouch.Alpha = 255;
                thisButton.SetBackground(gradient);
            }
```

```
            //thisButton.SetBackgroundColor(Android.Graphics.Color.Blue);
        }
    }



    protected override void Dispose(bool disposing)
    {
        if (thisButton != null)
        {
            thisButton.Touch -= ThisButton_Touch;
           // thisButton.Click -= HandleButtonClicked;
        }
        base.Dispose(disposing);
    }
  }
}

/*public enum Orientation {
    *//** draw the gradient from the top to the bottom *//*
    TOP_BOTTOM,
    *//** draw the gradient from the top-right to the bottom-left *//*
    TR_BL,
    *//** draw the gradient from the right to the left *//*
    RIGHT_LEFT,
    *//** draw the gradient from the bottom-right to the top-left *//*
    BR_TL,
    *//** draw the gradient from the bottom to the top *//*
    BOTTOM_TOP,
    *//** draw the gradient from the bottom-left to the top-right *//*
    BL_TR,
    *//** draw the gradient from the left to the right *//*
    LEFT_RIGHT,
    *//** draw the gradient from the top-left to the bottom-right *//*
```

```
    TL_BR,
  }*/
```

NoBorderEntryRenderer:

```
using System;
using Android.Content;
using Android.Graphics;
using Android.Graphics.Drawables;
using PoMOST.Droid.Renderers;
using Xamarin.Forms;
using Xamarin.Forms.Platform.Android;

[assembly: ExportRenderer(typeof(Entry), typeof(NoBorderEntryRenderer))]
namespace PoMOST.Droid.Renderers
{
  public class NoBorderEntryRenderer : EntryRenderer
  {
    public NoBorderEntryRenderer(Context context) : base(context)
    {
    }

    protected override void OnElementChanged(ElementChangedEventArgs<Entry> e)
            {
                base.OnElementChanged(e);
                //Control?.SetBackgroundColor(Android.Graphics.Color.Transparent);

      if (Control == null || Element == null || e.OldElement != null) return;

      var ourCustomColorHere = Xamarin.Forms.Color.FromHex("#d0eeff").ToAndroid();
      var gradient = new GradientDrawable();
      gradient.SetColor(Android.Graphics.Color.White);
      //gradient.SetCornerRadius(5);
     // gradient.SetStroke(1, ourCustomColorHere);
```

```
            Control.SetBackground(gradient);
            Control.SetPadding(20,20,20,20);

                    }
    }
}
```

NoLineEditorRenderer:

```
using System;
using Android.Content;
using PoMOST.Droid.Renderers;
using Xamarin.Forms;
using Xamarin.Forms.Platform.Android;

[assembly: ExportRenderer(typeof(Editor), typeof(NoLineEditorRenderer))]
namespace PoMOST.Droid.Renderers
{
    public class NoLineEditorRenderer : EditorRenderer
    {
        public NoLineEditorRenderer(Context context) : base(context)
        {
        }

        protected override void OnElementChanged(ElementChangedEventArgs<Editor> e)
        {
            base.OnElementChanged(e);
            Control?.SetBackgroundColor(Android.Graphics.Color.Transparent);
        }
    }
}
```

NoScrollBarListViewRenderer:

```
using System;
using PoMOST.Droid.Renderers;
using Xamarin.Forms;
using Xamarin.Forms.Platform.Android;

[assembly: ExportRenderer(typeof(ListView), typeof(NoScrollBarListViewRenderer))]
namespace PoMOST.Droid.Renderers
{
   public class NoScrollBarListViewRenderer : ListViewRenderer
   {
      protected override void OnElementChanged(ElementChangedEventArgs<ListView> e)
      {
         base.OnElementChanged(e);

         if (e.OldElement != null || Element == null)
         {
            return;
         }

         Control.VerticalScrollBarEnabled = false;

         Control.SetSelector(Android.Resource.Color.Transparent);
         Control.CacheColorHint = Xamarin.Forms.Color.Transparent.ToAndroid();



         //SetLayerType(Android.Views.LayerType.Hardware, null);

         //LayerDrawable layerDrawable = new LayerDrawable(new Drawable[] { backgroudDrawable });
         //layerDrawable.SetLayerInset(0, 7, 7, 7, 7);
         //SetBackgroundDrawable(layerDrawable);
      }
   }
}
```

TouchButton:

```
using System;
using Android.Content;
using PoMOST.Droid.Renderers;
using PoMOST.Views.Controls;
using Xamarin.Forms;
using Xamarin.Forms.Platform.Android;

[assembly: ExportRenderer(typeof(TouchContentView), typeof(TouchButton))]
namespace PoMOST.Droid.Renderers
{
   public class TouchButton : ViewRenderer<TouchContentView, Android.Views.View>
   {
      public TouchButton()
      {
         Console.WriteLine("INITTTT");
      }

      protected override void OnElementChanged(ElementChangedEventArgs<TouchContentView>
e)
      {
         Console.WriteLine("SET");
         base.OnElementChanged(e);


         this.Touch +=  (object sender, TouchEventArgs e2) =>
         {
            Console.WriteLine("test");
         };

      }
   }
```

```
}
```

Services:

SpeechToTextService:

```
using System;
using Android.App;
using Android.Bluetooth;
using Android.Content;
using Android.Speech;
using PoMOST.Droid;
using PoMOST.Droid.Services;
using PoMOST.Services;
using Xamarin.Forms;

[assembly: Xamarin.Forms.Dependency(typeof(SpeechToTextService))]
namespace PoMOST.Droid.Services
{
   public class SpeechToTextService : ISpeechToText
   {
      public SpeechToTextService()
      {
      }

      public event EventHandler<string> OnRecognize;

      public void StartRecognize()
      {
         Console.WriteLine("Start recognize");
         string rec = Android.Content.PM.PackageManager.FeatureMicrophone;
         if (rec != "android.hardware.microphone")
         {
            var alert = new AlertDialog.Builder(Forms.Context);
```

```
            alert.SetTitle("You don't seem to have a microphone to record with");
            alert.SetPositiveButton("OK", (sender, e) =>
            {
                return;
            });
            alert.Show();
        }

        var activity = (MainActivity)Forms.Context;
        var listener = new ActivityResultListener(activity, this);

        var voiceIntent = new Intent(RecognizerIntent.ActionRecognizeSpeech);
        voiceIntent.PutExtra(RecognizerIntent.ExtraLanguageModel,
RecognizerIntent.LanguageModelFreeForm);
        voiceIntent.PutExtra(RecognizerIntent.ExtraSpeechInputCompleteSilenceLengthMillis,
1500);

voiceIntent.PutExtra(RecognizerIntent.ExtraSpeechInputPossiblyCompleteSilenceLengthMillis,
1500);
        voiceIntent.PutExtra(RecognizerIntent.ExtraSpeechInputMinimumLengthMillis, 15000);
        voiceIntent.PutExtra(RecognizerIntent.ExtraMaxResults, 1);
        voiceIntent.PutExtra(RecognizerIntent.ExtraLanguage, Java.Util.Locale.Default);
        activity.StartActivityForResult(voiceIntent, 0);
    }

    private class ActivityResultListener
    {
        //private TaskCompletionSource<bool> Complete = new TaskCompletionSource<bool>();
        //public Task<bool> Task { get { return this.Complete.Task; } }

        MainActivity activity;
        SpeechToTextService parentService;
        public ActivityResultListener(MainActivity activity, SpeechToTextService parentService)
        {
            // subscribe to activity results
```

```
    this.activity = activity;
    this.activity.ActivityResult += OnActivityResult;
    this.parentService = parentService;
}

private void OnActivityResult(int requestCode, Result resultCode, Intent data)
{
    // unsubscribe from activity results
    if (requestCode == 0)
    {
        if (resultCode == Result.Ok)
        {
            activity.ActivityResult -= OnActivityResult;
            var matches = data.GetStringArrayListExtra(RecognizerIntent.ExtraResults);

            if (matches.Count != 0)
            {
                Console.WriteLine(matches[0]);
                parentService.OnRecognize?.Invoke(parentService, matches[0]);
                //string textInput = textBox.Text + matches[0];
                //textBox.Text = textInput;
                /*switch (matches[0].Substring(0, 5).ToLower())
                {
                    case "north":
                        //MovePlayer(0);
                        break;
                    case "south":
                        //MovePlayer(1);
                        break;
                }*/
            }
        }
        else
        {
```

```
            }
        }
    }
  }
 }
}
```

SPLASH

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;
using FFImageLoading;
using FFImageLoading.Svg.Platform;
using FFImageLoading.Views;

namespace PoMOST.Droid
{
  [Activity(Theme = "@style/splashTheme", MainLauncher = true, NoHistory = true)]
  public class Splash : Activity
  {
    protected override void OnCreate(Bundle savedInstanceState)
    {
      base.OnCreate(savedInstanceState);
```

```
        var imageView = new ImageViewAsync(this);

        ImageService.Instance
                .LoadCompiledResource("bb")
        .WithCustomDataResolver(new SvgDataResolver(0, 0, true))
        .Into(imageView);

        RelativeLayout rel = new RelativeLayout(this);
        rel.SetGravity(GravityFlags.Center);

        rel.AddView(imageView);
        SetContentView(rel);

        // Create your application here

        // StartActivity(typeof(MainActivity));
    }

    protected override void OnResume()
    {
        base.OnResume();
        Task startupWork = new Task(() => { SimulateStartup(); });
        startupWork.Start();
    }

    async void SimulateStartup()
    {
        await Task.Delay(2000); // Simulate a bit of startup work.
        StartActivity(new Intent(Application.Context, typeof(MainActivity)));
    }
  }
}
```

TwinTechs:

Droid:

Controls:

PageViewContainerRenderer:

```
using System;
using Xamarin.Forms;
using Xamarin.Forms.Platform.Android;
using TwinTechs.Controls;
using TwinTechs.Droid.Controls;
using Android.Views;
using TwinTechs.Droid.Extensions;
using TwinTechs.Extensions;
using Android.App;
using Android;
using System.Reflection;

[assembly: ExportRenderer (typeof(PageViewContainer), typeof(PageViewContainerRenderer))]
namespace TwinTechs.Droid.Controls
{
    public class PageViewContainerRenderer : ViewRenderer<PageViewContainer,Android.Views.View>
    {
        public PageViewContainerRenderer ()
        {

        }

        Page _currentPage;

        protected override void OnElementChanged (ElementChangedEventArgs<PageViewContainer> e)
        {
```

```
                base.OnElementChanged (e);
                var pageViewContainer = e.NewElement as PageViewContainer;
                if (e.NewElement != null) {
                        ChangePage (e.NewElement.Content);
                } else {
                        ChangePage (null);
                }

        }

        protected override void OnElementPropertyChanged (object sender,
System.ComponentModel.PropertyChangedEventArgs e)
        {
                base.OnElementPropertyChanged (sender, e);
                if (e.PropertyName == "Content") {
                        ChangePage (Element.Content);
                }
        }

        bool _contentNeedsLayout;

        protected override void OnLayout (bool changed, int l, int t, int r, int b)
        {
                base.OnLayout (changed, l, t, r, b);
                if ((changed || _contentNeedsLayout) && this.Control != null) {
                        if (_currentPage != null) {
                                _currentPage.Layout (new Rectangle (0, 0, Element.Width,
Element.Height));
                        }
                        var msw = MeasureSpec.MakeMeasureSpec (r - l,
MeasureSpecMode.Exactly);
                        var msh = MeasureSpec.MakeMeasureSpec (b - t,
MeasureSpecMode.Exactly);
                        this.Control.Measure (msw, msh);
                        this.Control.Layout (0, 0, r, b);
```

```
                _contentNeedsLayout = false;
        }
}

private int ConvertPixelsToDp (float pixelValue)
{
        var dp = (int)((pixelValue) / Resources.DisplayMetrics.Density);
        return dp;
}


void ChangePage (Page page)
{

        //TODO handle current page
        if (page != null) {
                var parentPage = Element.GetParentPage ();
                page.Parent = parentPage;

                var existingRenderer = page.GetRenderer ();
                if (existingRenderer == null) {
                        var renderer = RendererFactory.GetRenderer (page);
                        page.SetRenderer (renderer);
                        existingRenderer = page.GetRenderer ();
                }
                _contentNeedsLayout = true;
                SetNativeControl (existingRenderer.ViewGroup);
                Invalidate ();
                //TODO update the page
                _currentPage = page;
        } else {
                //TODO - update the page
                _currentPage = null;
        }
```

```
                    if (_currentPage == null) {
                        //have to set somethign for android not to get pissy
                        var view = new Android.Views.View (this.Context);
                        view.SetBackgroundColor      (Element.BackgroundColor.ToAndroid
());
                        SetNativeControl (view);
                    }
                }

        }
}
```

ViewMaskExtensionProvider:

```
using System;
using TwinTechs.Controls;
using System.Drawing;
using System.Runtime.InteropServices;
using Xamarin.Forms;
using TwinTechs.Droid.Extensions;
using Android.Widget;
using Android.Graphics;
using Android.Views;

namespace TwinTechs.Droid.Controls
{
    public class ShadowWrapper : Android.Widget.AbsoluteLayout
    {

        Paint paint;

        public ViewGroup Content { get; private set; }
```

```
        ViewGroup OriginalParent { get; set; }

        public ShadowWrapper (Android.Content.Context context, ViewGroup realContent) :
base (context)
        {
            Content = realContent;
            OriginalParent = realContent.Parent as ViewGroup;
            OriginalParent.RemoveView (realContent);
            this.SetMinimumHeight (realContent.Height);
            this.SetMinimumWidth (realContent.Width);
            this.AddView (realContent, new LinearLayout.LayoutParams
(LinearLayout.LayoutParams.FillParent,
                LinearLayout.LayoutParams.FillParent));
            OriginalParent.AddView (this, new LinearLayout.LayoutParams
(LinearLayout.LayoutParams.FillParent,
                LinearLayout.LayoutParams.FillParent));
            this.SetBackgroundColor (Android.Graphics.Color.Red);

        }

        protected override void OnMeasure (int widthMeasureSpec, int heightMeasureSpec)
        {
            base.OnMeasure (widthMeasureSpec, heightMeasureSpec);
        }

        public void ResetParent ()
        {
            var content = Content;
            this.RemoveView (content);
            OriginalParent.AddView (content, new LinearLayout.LayoutParams
(LinearLayout.LayoutParams.FillParent,
                LinearLayout.LayoutParams.FillParent));
        }
```

```
    }

    public class ViewMaskExtensionProvider : IViewEffectExtensionProvider
    {
        public ViewMaskExtensionProvider ()
        {
        }

        #region IViewMaskExtensionProvider implementation

        public void ApplyMaskToView (Xamarin.Forms.View view, ViewMaskerType maskType)
        {
            //we need to wrap the view inside another view
        }


        public void ToggleViewShadow (Xamarin.Forms.View view, bool isOn)
        {
            var renderer = view.GetOrCreateRenderer ();
            var nativeView = renderer.ViewGroup;
            var shadowWrapper = nativeView.Parent as ShadowWrapper;

            if (shadowWrapper != null) {
                //move back to original parent
                shadowWrapper.ResetParent ();
                view.SizeChanged -= OnSizeChanged;
            }
            if (isOn) {
                shadowWrapper = new ShadowWrapper (nativeView.Context, nativeView);
                view.SizeChanged += OnSizeChanged;
            }
        }
```

```
            #endregion

            void OnSizeChanged (object sender, EventArgs e)
            {

                var view = sender as Xamarin.Forms.View;
                var renderer = view.GetOrCreateRenderer ();
                var nativeView = renderer.ViewGroup;
                var shadowWrapper = nativeView.Parent as ShadowWrapper;
                if (shadowWrapper != null) {

                    shadowWrapper.Layout (0, 0, (int)view.Width, (int)view.Height);
                }
            }
        }
}
```

Extensions:

ViewExtensions:

```
using System;
using Xamarin.Forms;
using Xamarin.Forms.Platform.Android;
using System.Reflection;

namespace TwinTechs.Droid.Extensions
{
    public static class ViewExtensions
    {
        private static readonly Type _platformType = Type.GetType
("Xamarin.Forms.Platform.Android.Platform, Xamarin.Forms.Platform.Android", true);
        private static BindableProperty _rendererProperty;
```

```
        public static BindableProperty RendererProperty {
            get {
                _rendererProperty = (BindableProperty)_platformType.GetField
("RendererProperty", BindingFlags.Public | BindingFlags.NonPublic | BindingFlags.Static)
                    .GetValue (null);

                return _rendererProperty;
            }
        }

        public static IVisualElementRenderer GetRenderer (this BindableObject
bindableObject)
        {
            var value = bindableObject.GetValue (RendererProperty);
            return (IVisualElementRenderer)bindableObject.GetValue
(RendererProperty);
        }

        public static Android.Views.View GetNativeView (this BindableObject
bindableObject)
        {
            var renderer = bindableObject.GetRenderer ();
            var viewGroup = renderer.ViewGroup;
            return viewGroup;
        }

        public static void SetRenderer (this BindableObject bindableObject,
IVisualElementRenderer renderer)
        {
//          var value = bindableObject.GetValue (RendererProperty);
            bindableObject.SetValue (RendererProperty, renderer);
        }

        public static Point GetNativeScreenPosition (this BindableObject bindableObject)
        {
```

```
                var view = bindableObject.GetNativeView ();
                var point = Point.Zero;
                if (view != null) {
                        int[] location = new int[2];
                        view.GetLocationOnScreen (location);
                        point = new Xamarin.Forms.Point (location [0], location [1]);
                }
                return point;
        }

        /// <summary>
        /// Gets the or create renderer.
        /// </summary>
        /// <returns>The or create renderer.</returns>
        /// <param name="source">Source.</param>
        public static IVisualElementRenderer GetOrCreateRenderer (this VisualElement source)
        {
                var renderer = source.GetRenderer ();
                if (renderer == null) {
                        renderer = RendererFactory.GetRenderer (source);
                        source.SetRenderer (renderer);
                        renderer = source.GetRenderer ();
                }
                return renderer;
        }
    }
}
```

POMOST:

ContainerAccossor:

```
using System;
```

```
using Autofac;
using PoMOST.Services;
using PoMOST.Services.Navigation;
using PoMOST.ViewModels;
using PoMOST.ViewModels.CellsViewModels;
using PoMOST.ViewModels.PopupsViewModels;
using PoMOST.ViewModels.RegistrationViewModels;
using PoMOST.ViewModels.RegistrationViewModels.RegistrationPopupsViewModels;
using Xamarin.Forms;

namespace PoMOST
{
   public static class ContainerAccessor
   {
      private static IContainer container;

      public static void RegisterDependencies()
      {
         var builder = new ContainerBuilder();

         //Register services

//builder.RegisterType<DummyHelpRequestsService>().As<IHelpRequestsService>().SingleInstance();

builder.RegisterType<ServerHelpRequestsService>().As<IHelpRequestsService>().SingleInstance();
         builder.RegisterType<UserService>().As<IUserService>().SingleInstance();
         builder.RegisterType<ServerAuthService>().As<IAuthService>().SingleInstance();
         builder.RegisterType<UploadService>().As<IUploadService>().SingleInstance();

         //Register helpers
         builder.RegisterType<DisplayErrorService>().SingleInstance();

builder.RegisterInstance(DependencyService.Get<ISpeechToText>()).As<ISpeechToText>().SingleInstance();
```

```
        //Register navigation
        builder.RegisterInstance(new
NavigationService(Xamarin.Forms.Application.Current)).As<INavigationService>();

        //Register Pages ViewModels
        builder.RegisterType<MainPageViewModel>();
        builder.RegisterType<HelpRequestsPageViewModel>();
        builder.RegisterType<VoiceCommandsPageViewModel>();
        builder.RegisterType<SettingsPageViewModel>();
        builder.RegisterType<AddHelpRequestPageViewModel>();
        builder.RegisterType<ProfilePageViewModel>();
        builder.RegisterType<ProfilePageViewModel>();
        builder.RegisterType<AuthorsPageViewModel>();
        builder.RegisterType<NamePageViewModel>();
        builder.RegisterType<PhoneNumberPageViewModel>();
        builder.RegisterType<TermsPageViewModel>();
        builder.RegisterType<GeolocationPageViewModel>();
        builder.RegisterType<PhotoPageViewModel>();
        builder.RegisterType<RegistrationPageViewModel>();
        builder.RegisterType<SuccessPageViewModel>();

        //Register Popups ViewModel
        builder.RegisterType<ShopRequestPopupViewModel>();
        builder.RegisterType<HealthRequestPopupViewModel>();
        builder.RegisterType<TransportRequestPopupViewModel>();
        builder.RegisterType<PostRequestPopupViewModel>();
        builder.RegisterType<AlarmRequestPopupViewModel>();
        builder.RegisterType<InternetLearnRequestPopupViewModel>();
        builder.RegisterType<OtherRequestPopupViewModel>();
        builder.RegisterType<VoiceConfirmPopupViewModel>();
        builder.RegisterType<SexPopupViewModel>();
        builder.RegisterType<UserPopupViewModel>();
        builder.RegisterType<PhotoPopupViewModel>();
```

```
            builder.RegisterType<HelpRequestPopupViewModel>();

            //Register Cells ViewModel
            builder.RegisterType<HelpRequestCellViewModel>();

            container = builder.Build();
        }

        public static T Resolve<T>()
        {
            return container.Resolve<T>();
        }

        public static T Resolve<T>(NamedParameter parameter)
        {
            return container.Resolve<T>(parameter);
        }
    }
}
```

Exceptions:

HttpResponseStatusException:

```
using System;
using System.Net;
using System.Net.Http;

namespace PoMOST.Exceptions
{
    public class HttpResponseStatusException : HttpRequestException
    {
        public HttpStatusCode StatusCode { get; set; }
```

```
    }
}
```

EnauthorizedException:

```
using System;
namespace PoMOST.Exceptions
{
    public class UnauthorizedException : Exception
    {
        public UnauthorizedException()
        {
        }
    }
}
```

Helpers:

BoolConverter:

```
using System;
using System.Globalization;
using Xamarin.Forms;

namespace PoMOST.Helpers
{
    public class BoolConverter : IValueConverter
    {


        public object Convert(object value, Type targetType, object parameter, CultureInfo culture)
        {
            var valueAsBool = (bool)value;
```

```
            return !valueAsBool;
        }

        public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)
        {
            throw new NotImplementedException();
        }
    }
}
```

BorderColorConverter:

```
using System;
using System.Globalization;
using PoMOST.Models;
using PoMOST.Styles;
using Xamarin.Forms;

namespace PoMOST.Helpers
{
    public class BorderColorConverter : IValueConverter
    {
        public object Convert(object value, Type targetType, object parameter, CultureInfo culture)
        {
            var request = (HelpRequest.Categories)value;

            if (request == HelpRequest.Categories.ALERT)
                return Color.Red;
            else return CustomStyles.LightBlue;
        }

        public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)
        {
```

```
            throw new NotImplementedException();
        }
    }
}
```

ObservableCollectionFast:

```
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Collections.Specialized;
using System.ComponentModel;

namespace PoMOST
{
    public class ObservableCollectionFast<T> : ObservableCollection<T>
    {
        public ObservableCollectionFast() : base() { }

        public ObservableCollectionFast(IEnumerable<T> collection) : base(collection) {
}

        public ObservableCollectionFast(List<T> list) : base(list) { }

        public void AddRange(IEnumerable<T> range)
        {
            foreach (var item in range)
            {
                Items.Add(item);
            }

            this.OnPropertyChanged(new PropertyChangedEventArgs("Count"));
            this.OnPropertyChanged(new PropertyChangedEventArgs("Item[]"));
        this.OnCollectionChanged(new
NotifyCollectionChangedEventArgs(NotifyCollectionChangedAction.Reset));
```

```
                    //this.OnCollectionChanged(new
NotifyCollectionChangedEventArgs(NotifyCollectionChangedAction.Add,range,startIndex));
                }

                public void Reset(IEnumerable<T> range)
                {
                    this.Items.Clear();

                    AddRange(range);
                }
        }
}
```

Models:

ErrorMsg:

```
using System;
namespace PoMOST.Models
{
   public class ErrorMsg
   {
      public string Text { get; set; }
      public bool Success { get; set; }
   }
}
```

HelpRequest:

```
using System;
using System.Collections.Generic;
using Newtonsoft.Json;

namespace PoMOST.Models
```

```
{
    public class HelpRequest
    {
        public User Author { get; set; }
        public int Distance {get; set; }
        public DateTime CreationDate { get; set; }
        public int Id { get; set; }
        public Categories Category { get; set; }
        public string Text { get; set; }
        public Location Location { get; set; }

        [JsonIgnore]
        public string CategoryText
        {
            get
            {
                switch(Category)
                {
                    case Categories.ALERT:
                        return "Alarm";
                        break;
                    case Categories.HEALTH:
                        return "Zdrowie";
                        break;
                    case Categories.IT:
                        return "Internet i komputer";
                        break;
                    case Categories.MAIL:
                        return "Poczta";
                        break;
                    case Categories.SHOPPING:
                        return "Zakupy";
                        break;
                    case Categories.TRANSPORT:
```

```
                return "Transport";
                break;
            case Categories.OTHER:
                return "Inne";
                break;
            default:
                return "";
                break;

        }
      }
    }

    public enum Categories
    {
      HEALTH,
      IT,
      TRANSPORT,
      MAIL,
      SHOPPING,
      ALERT,
      OTHER
    };

  }
}
```

HelpRequestDto:

```
using System;
using Newtonsoft.Json;
using static PoMOST.Models.HelpRequest;

namespace PoMOST.Models
```

```
{
    public class HelpRequestDto
    {
        [JsonProperty("category")]
        public string Category { get; set; }

        [JsonProperty("location")]
        public Location Location { get; set; }

        [JsonProperty("text")]
        public string Text { get; set; }
    }
}
```

Location:

```
using System;
using Newtonsoft.Json;

namespace PoMOST.Models
{
    public class Location
    {
        [JsonProperty("latitude")]
        public double Latitude { get; set; }

        [JsonProperty("longitude")]
        public double Longitude { get; set; }
    }
}
```

PageResult:

```
using System;
```

```
using System.Collections.Generic;

namespace PoMOST.Models
{
    public class PageResult<T>
    {
        public IList<T> Content { get; set; }
        public bool First { get; set; }
        public bool Last { get; set; }
        public int Number { get; set; }
        public int NumberOfElements { get; set; }
        public int Size { get; set; }
        public int TotalElements { get; set; }
        public int TotalPages { get; set; }
    }
}
```

ParseToken:

```
using System;
namespace PoMOST.Models
{
    public class ParseToken
    {
        public int uid { get; set; }
    }
}
```

PostUserDto:

```
using System;
using Newtonsoft.Json;

namespace PoMOST.Models
```

```
{
    public class PostUserDto
    {
        [JsonProperty("avatar")]
        public string Avatar { get; set; }

        [JsonProperty("gender")]
        public string Gender { get; set; }

        [JsonProperty("helper")]
        public bool Helper { get; set; }

        [JsonProperty("location")]
        public Location Location { get; set; }

        [JsonProperty("name")]
        public string Name { get; set; }

        [JsonProperty("needy")]
        public bool Needy { get; set; }

        [JsonProperty("phone")]
        public string Phone { get; set; }
    }
}
```

Token:

```
using System;
namespace PoMOST.Models
{
    public class Token
    {
        public string Value { get; set; }
```

```
    }
}
```

UpdateUserDto:

```
using System;
using Newtonsoft.Json;

namespace PoMOST.Models
{
    public class UpdateUserDto
    {
        [JsonProperty("avatar")]
        public string Avatar { get; set; }

        [JsonProperty("description")]
        public string Description { get; set; }

        [JsonProperty("gender")]
        public string Gender { get; set; }

        [JsonProperty("helper")]
        public bool Helper { get; set; }

        [JsonProperty("name")]
        public string Name { get; set; }

        [JsonProperty("needy")]
        public bool Needy { get; set; }

        [JsonProperty("notificationActive")]
        public bool NotificationActive { get; set; }

        [JsonProperty("notificationRadius")]
```

```
        public int NotificationRadius { get; set; }

        [JsonProperty("phone")]
        public string Phone { get; set; }
    }
}
```

UploadData:

```
using System;
using Newtonsoft.Json;

namespace PoMOST.Models
{
    public class UploadData
    {
        [JsonProperty("fileName")]
        public string FileName { get; set; }

        [JsonProperty("format")]
        public string Format { get; set; }

        [JsonProperty("id")]
        public int Id { get; set; }

        [JsonProperty("kBfileSize")]
        public int KBfileSize { get; set; }

        [JsonProperty("resourceHttpLink")]
        public string ResourceHttpLink { get; set; }

        [JsonProperty("uploadType")]
        public string UploadType { get; set; }
    }
```

```
}
```

User:

```
using System;
using Newtonsoft.Json;

namespace PoMOST.Models
{
    public class User
    {
        [JsonProperty("avatar")]
        public string Avatar { get; set; }

        [JsonIgnore]
        public string AvatarSource
        {
            get
            {
                if (string.IsNullOrEmpty(Avatar))
                {
                    var role = (Needy) ? "senior" : "helper";
                    return "avatar_" + role + "_" + Gender;
                }
                return Avatar;
            }
        }

        [JsonProperty("description")]
        public string Description { get; set; }

        [JsonProperty("gender")]
        public string Gender { get; set; }
```

```
        [JsonProperty("helper")]
        public bool Helper { get; set; }

        [JsonProperty("id")]
        public int Id { get; set; }

        [JsonProperty("location")]
        public Location Location { get; set; }

        [JsonProperty("name")]
        public string Name { get; set; }

        [JsonProperty("needy")]
        public bool Needy { get; set; }

        [JsonProperty("notificationRadius")]
        public int NotificationRadius { get; set; }

        [JsonProperty("notificationsActive")]
        public bool NotificationsActive { get; set; }

        [JsonProperty("phone")]
        public string Phone { get; set; }
    }
}
```

PoMost:

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using FFImageLoading.Svg.Forms;
using Plugin.FirebasePushNotification;
using Plugin.SecureStorage;
```

```
using PoMOST.Services;
using PoMOST.Views.Pages;
using PoMOST.Views.Pages.Registration;
using TwinTechs.Controls;
using Xamarin.Forms;

namespace PoMOST
{
    public class App : Application
    {
        public static float ScreenWidth;
        public static float ScreenHeight;
        public static readonly string BaseAddress = "http://lukskar.pl:8088";
        public static string Token = null;
        public static int UserID = 0;
        public static string ROLE = "BOOTH";

        IUserService userService;

        public App()
        {
            // The root page of your application
            //CrossSecureStorage.Current.DeleteKey("Token");

            if(CrossSecureStorage.Current.HasKey("Token"))
            {
                if (CrossSecureStorage.Current.HasKey("UserID"))
                {
                    App.Token = CrossSecureStorage.Current.GetValue("Token");
                    App.UserID = int.Parse(CrossSecureStorage.Current.GetValue("UserID"));
                    if (CrossSecureStorage.Current.HasKey("UserRole"))
                    {
                        App.ROLE = CrossSecureStorage.Current.GetValue("UserRole");
                    }
```

```
        }
        else
        {
            CrossSecureStorage.Current.DeleteKey("Token");
        }
    }

    ContainerAccessor.RegisterDependencies();

    var authService = ContainerAccessor.Resolve<IAuthService>();
    userService = ContainerAccessor.Resolve<IUserService>();

    if (string.IsNullOrEmpty(App.Token))
    {
        MainPage = new NavigationPage(new RegistrationPage());
    }
    else
    {
        MainPage = new MainPage();
    }
}

protected override void OnStart()
{
    CrossFirebasePushNotification.Current.OnTokenRefresh += Current_OnTokenRefresh;
}

protected override void OnSleep()
{
    // Handle when your app sleeps
}

protected override void OnResume()
{
```

```
            // Handle when your app resumes
        }

        async void Current_OnTokenRefresh(object source,
Plugin.FirebasePushNotification.Abstractions.FirebasePushNotificationTokenEventArgs e)
        {
            try
            {
                Debug.WriteLine("TOKEN "+e.Token);
                if (App.Token != null)
                {
                    await userService.UpdatePushNotificationToken(e.Token);
                }
            }
            catch (Exception exc)
            {
                Debug.WriteLine(exc.Message);
            }
        }
    }
}
```

Services:

DisplayErrorService:

```
using System;
using PoMOST.Models;

namespace PoMOST.Services
{
    public class DisplayErrorService
    {
        public event EventHandler<ErrorMsg> OnError;
```

```
        public DisplayErrorService()
        {

        }

        public void HandleError(string errorMsg, bool isSuccess = false)
        {
            OnError?.Invoke(this, new ErrorMsg(){Text=errorMsg, Success= isSuccess});
        }
    }
}
```

IAuthService:

```
using System;
using System.Threading.Tasks;
using PoMOST.Models;

namespace PoMOST.Services
{
    public interface IAuthService
    {
        Task<bool> Login();
    }
}
```

IHelpRequestsService:

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using PoMOST.Models;
```

```
namespace PoMOST.Services
{
    public interface IHelpRequestsService
    {
        Task<IList<HelpRequest>> GetHelpRequests(Location location);
        Task<IList<HelpRequest>> GetMyHelpRequests();
        Task<HelpRequest> CreateHelpRequest(HelpRequestDto requestData);
        ObservableCollectionFast<HelpRequest> Requests { get; set; }
        ObservableCollectionFast<HelpRequest> MyRequests { get; set; }
        Task<bool> DeleteHelpRequest(int id);
    }
}
```

ISpeechToText:

```
using System;
namespace PoMOST.Services
{
    public interface ISpeechToText
    {
        event EventHandler<string> OnRecognize;
        void StartRecognize();
    }
}
```

IUploadService:

```
using System;
using System.IO;
using System.Threading.Tasks;
using PoMOST.Models;

namespace PoMOST.Services
{
```

```
    public interface IUploadService
    {
        Task<UploadData> UploadFile(Stream fileStream);
    }
}
```

IUserSevice:

```
using System;
using System.Threading.Tasks;
using PoMOST.Models;

namespace PoMOST.Services
{
    public interface IUserService
    {
        Task<bool> UpdatePushNotificationToken(string token);
        Task<User> RegisterUser(PostUserDto userData);
        Task<User> GetUserById(int id, string token);
        Task<bool> UpdateUser(int id, User userData, string token);
        Task<bool> UpdateLocation(double longitude, double latitude);
    }
}
```

Navigation:

RestClient:

```
using System;
using System.Diagnostics;
using System.Net;
using System.Net.Http;
using System.Threading.Tasks;
using PoMOST.Exceptions;
```

```
namespace PoMOST.Services
{
   public class RestClient
   {
      private HttpClient client;
      public RestClient(string baseAddress)
      {
         client = new HttpClient();
         client.BaseAddress = new Uri(baseAddress);
         client.DefaultRequestHeaders.Add("Accept", "application/json");
      }

      public async Task<T> GetAsync<T>(string path, string token = null)
      {
         try
         {
            var request = new HttpRequestMessage(HttpMethod.Get, path);

            if(token!=null)
               request.Headers.Add("X-Authorization", token);

            using (var result = await client.SendAsync(request))
            {
               try
               {
                  result.EnsureSuccessStatusCode();
                  var json = await result.Content.ReadAsStringAsync();
                  return Newtonsoft.Json.JsonConvert.DeserializeObject<T>(json);
               }
               catch (HttpRequestException ex)
               {
                  if (result.StatusCode == HttpStatusCode.Unauthorized)
                  {
```

```
                throw new UnauthorizedException();
            }
            else
            {
                return default(T);
            }
        }
    }
    catch (Exception e)
    {
        throw e;
    }
}

public async Task<bool> DeleteAsync(string path, string token = null)
{
    try
    {
        var request = new HttpRequestMessage(HttpMethod.Delete, path);

        if (token != null)
            request.Headers.Add("X-Authorization", token);

        using (var result = await client.SendAsync(request))
        {
            try
            {
                result.EnsureSuccessStatusCode();
                return true;
            }
            catch (HttpRequestException ex)
            {
                if (result.StatusCode == HttpStatusCode.Unauthorized)
```

```
                {
                    throw new UnauthorizedException();
                }
                else
                {
                    return false;
                }
            }
        }
    }
    catch (Exception e)
    {
        throw e;
    }
}
```

```
public async Task<HttpResponseMessage> PostAsync(string path, StringContent content, string token = null)
{
    try
    {
        var request = new HttpRequestMessage(HttpMethod.Post, path);

        if (token != null)
            request.Headers.Add("X-Authorization", token);

        request.Content = content;

        var result = await client.SendAsync(request);
        result.EnsureSuccessStatusCode();
        return result;
    }
    catch(Exception e)
    {
```

```
            throw e;
        }
    }

    public async Task<HttpResponseMessage> PostAsync(string path, MultipartFormDataContent content, string token = null)
    {
        try
        {
            var request = new HttpRequestMessage(HttpMethod.Post, path);

            if (token != null)
                request.Headers.Add("X-Authorization", token);

            request.Content = content;

            var result = await client.SendAsync(request);
            result.EnsureSuccessStatusCode();
            return result;
        }
        catch (Exception e)
        {
            throw e;
        }
    }

    public async Task<HttpResponseMessage> PutAsync(string path, StringContent content, string token = null)
    {
        try
        {
            var request = new HttpRequestMessage(HttpMethod.Put, path);

            if (token != null)
                request.Headers.Add("X-Authorization", token);
```

```
            request.Content = content;
            var result = await client.SendAsync(request);
            result.EnsureSuccessStatusCode();
            return result;
        }
        catch (Exception e)
        {
            throw e;
        }
    }
  }
}
```

ServerAuthService:

```
using System;
using System.Diagnostics;
using System.Net.Http;
using System.Text;
using System.Threading.Tasks;
using Plugin.FirebasePushNotification;
using Plugin.SecureStorage;
using PoMOST.Models;

namespace PoMOST.Services
{
  public class ServerAuthService : IAuthService
  {
    RestClient client;
    IUserService userService;

    public ServerAuthService()
    {
```

```
        this.userService = userService;

        client = new RestClient($"{App.BaseAddress}/api/");
    }

    async public Task<bool> Login()
    {
        try
        {
            string phone = null;
            if(CrossSecureStorage.Current.HasKey("PhoneNumber"))
            {
                Debug.WriteLine("Logowanie2");
                phone = CrossSecureStorage.Current.GetValue("PhoneNumber");
            }
            if (phone == null)
            {
                Debug.WriteLine("Logowanie3");
                return false;
            }

            var loginData = new
            {
                login = phone
            };
            Debug.WriteLine("Logowanie1");
            var jsonToSend = Newtonsoft.Json.JsonConvert.SerializeObject(loginData);
            var result = await client.PostAsync("auth/login", new StringContent(jsonToSend, Encoding.UTF8, "application/json"));
            var json = await result.Content.ReadAsStringAsync();
            var token = Newtonsoft.Json.JsonConvert.DeserializeObject<Token>(json);

            CrossSecureStorage.Current.SetValue("Token", token.Value);
            App.Token = token.Value;
```

```
            var clearToken = token.Value.Replace("Bearer ", "");

            var splitToken = clearToken.Split(new char[] { '.' });

            var str = JWT.JsonWebToken.Base64UrlDecode(splitToken[1]);

            var userToken = Newtonsoft.Json.JsonConvert.DeserializeObject<ParseToken>(Encoding.UTF8.GetString(str, 0, str.Length));

            CrossSecureStorage.Current.SetValue("UserID", userToken.uid.ToString());
            App.UserID = userToken.uid;

            return true;
        }
        catch (Exception e)
        {
            Debug.WriteLine(e.Message);
            return false;
        }

    }
  }
}
```

ServerHelpRequestsService:

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using PoMOST.Models;
using System.Net.Http;
using System.Diagnostics;
using PoMOST.Exceptions;
```

```
using System.Net;
using Plugin.Connectivity;
using System.Collections.ObjectModel;
using System.Text;
using System.Linq;
using Plugin.Geolocator;
using Plugin.FirebasePushNotification;
using Xamarin.Forms;

namespace PoMOST.Services.Navigation
{
   public class ServerHelpRequestsService : IHelpRequestsService
   {
      RestClient client;
      DisplayErrorService errorService;
      IAuthService authService;
      IUserService userService;

      public ObservableCollectionFast<HelpRequest> Requests { get; set; }
      public ObservableCollectionFast<HelpRequest> MyRequests { get; set; }

      public ServerHelpRequestsService(DisplayErrorService errorService, IAuthService authService, IUserService userService)
      {
         this.errorService = errorService;
         this.authService = authService;
         this.userService = userService;

         Requests = new ObservableCollectionFast<HelpRequest>();
         MyRequests = new ObservableCollectionFast<HelpRequest>();

         client = new RestClient($"{App.BaseAddress}/api/");

         CrossFirebasePushNotification.Current.OnNotificationReceived += (object source, Plugin.FirebasePushNotification.Abstractions.FirebasePushNotificationDataEventArgs e) =>
```

```
        {
            Device.BeginInvokeOnMainThread(() =>
            {
                errorService.HandleError("Dodano nową prośbę w Twojej oklolicy. Odśwież listę.", true);
            });

        };
    }

    async public Task<IList<HelpRequest>> GetHelpRequests(Location location)
    {
        if(!CrossConnectivity.Current.IsConnected)
        {
            errorService.HandleError("Nie masz połączenia z internetem.");
            return null;
        }
        if (!CrossGeolocator.Current.IsGeolocationEnabled)
        {
            errorService.HandleError("Włącz usługę lokalizacji.");
            return null;
        }
        try
        {
            var position = await CrossGeolocator.Current.GetPositionAsync(TimeSpan.FromSeconds(2));
            if(position!=null)
            {
                var updatePosition = await userService.UpdateLocation(position.Longitude, position.Latitude);
            }

            var result = await client.GetAsync<PageResult<HelpRequest>>("help/nearby?page=0,100", App.Token);
            Requests.Reset(result.Content);
```

```
            return result.Content;
        }
        catch(UnauthorizedException)
        {
            var login = await authService.Login();
            if (login)
            {
                return await GetHelpRequests(location);
            }
            else
            {
                errorService.HandleError("Wystąpiły problemy w aplikacji.");
                return null;
            }
        }
        catch (Exception)
        {
            throw;
        }
    }

    async public Task<IList<HelpRequest>> GetMyHelpRequests()
    {
        if (!CrossConnectivity.Current.IsConnected)
        {
            errorService.HandleError("Nie masz połączenia z internetem.");
            return null;
        }
        try
        {
            var result = await client.GetAsync<PageResult<HelpRequest>>("help/own?page=0,100",
App.Token);
            MyRequests.AddRange(result.Content);
            Debug.WriteLine("GetMyHelpRequests "+result.Content.Count);
```

```
            return result.Content;
        }
        catch (UnauthorizedException)
        {
            var login = await authService.Login();
            if (login)
            {
                return await GetMyHelpRequests();
            }
            else
            {
                errorService.HandleError("Wystąpiły problemy w aplikacji.");
                return null;
            }
        }
        catch (Exception)
        {
            throw;
        }
    }

    async public Task<HelpRequest> CreateHelpRequest(HelpRequestDto requestData)
    {
        if (!CrossConnectivity.Current.IsConnected)
        {
            errorService.HandleError("Nie masz połączenia z internetem.");
            return null;
        }
        try
        {
            var jsonToSend = Newtonsoft.Json.JsonConvert.SerializeObject(requestData);
            Debug.WriteLine(jsonToSend);
            var content = new StringContent(jsonToSend, Encoding.UTF8, "application/json");
            var result = await client.PostAsync("help", content, App.Token);
```

```
        if (result.StatusCode == System.Net.HttpStatusCode.OK)
        {
            var json = await result.Content.ReadAsStringAsync();
            Debug.WriteLine(json);
            var request = Newtonsoft.Json.JsonConvert.DeserializeObject<HelpRequest>(json);
            MyRequests.Insert(0,request);
            return request;
        }
        else
        {
            errorService.HandleError("Wystąpiły problemy w aplikacji.");
            return null;
        }
    }
    catch (Exception ex)
    {
        Debug.WriteLine("CreateHelpRequest" +ex.Message);
        errorService.HandleError("Wystąpiły problemy w aplikacji.");
        return null;
    }
}

async public Task<bool> DeleteHelpRequest(int id)
{
    if (!CrossConnectivity.Current.IsConnected)
    {
        errorService.HandleError("Nie masz połączenia z internetem.");
        return false;
    }
    try
    {
        var result = await client.DeleteAsync($"help/{id}", App.Token);
        if (result)
        {
```

```
                var item = MyRequests.Where(o => o.Id == id).FirstOrDefault();
                if (item!=null)
                {
                    MyRequests.Remove(item);
                    errorService.HandleError("Prośba została usunięta.", true);
                    return true;
                }
                else
                {
                    return false;
                }
            }
            else
            {
                errorService.HandleError("Wystąpiły problemy w aplikacji.");
                return false;
            }
        }
        catch (Exception ex)
        {
            Debug.WriteLine("CreateHelpRequest" + ex.Message);
            errorService.HandleError("Wystąpiły problemy w aplikacji.");
            return false;
        }
    }
  }
}
```

UploadService:

```
using System;
using System.IO;
using System.Net.Http;
using System.Net.Http.Headers;
```

```
using System.Threading.Tasks;
using Plugin.Connectivity;
using PoMOST.Models;

namespace PoMOST.Services
{
   public class UploadService : IUploadService
   {
      RestClient client;
      DisplayErrorService errorService;

      public UploadService(DisplayErrorService errorService)
      {
         this.errorService = errorService;
         client = new RestClient($"{App.BaseAddress}/api/");
      }

      async public Task<UploadData> UploadFile(Stream fileStream)
      {
         if (!CrossConnectivity.Current.IsConnected)
         {
            errorService.HandleError("Nie masz połączenia z internetem.");
            return null;
         }
         try
         {
            var form = new MultipartFormDataContent();
            var data = new StreamContent(fileStream);

            data.Headers.ContentDisposition = ContentDispositionHeaderValue.Parse("form-data");
            data.Headers.ContentDisposition.Parameters.Add(new   NameValueHeaderValue("name",
"file"));
            data.Headers.ContentDisposition.Parameters.Add(new
NameValueHeaderValue("filename", "test.jpg"));
```

```
            form.Add(data);

            var result = await client.PostAsync("uploads", form);
            var json = await result.Content.ReadAsStringAsync();
            var uploadData = Newtonsoft.Json.JsonConvert.DeserializeObject<UploadData>(json);
            return uploadData;
        }
        catch (Exception)
        {
            throw;
        }
    }
  }
}
```

UserService:

```
using System;
using System.Diagnostics;
using System.Net.Http;
using System.Text;
using System.Threading.Tasks;
using Plugin.Connectivity;
using Plugin.SecureStorage;
using PoMOST.Exceptions;
using PoMOST.Models;

namespace PoMOST.Services
{
  public class UserService : IUserService
  {
    RestClient client;
    IAuthService authService;
    DisplayErrorService errorService;
```

```
public UserService(IAuthService authService, DisplayErrorService errorService)
{
    this.authService = authService;
    this.errorService = errorService;
    client = new RestClient($"{App.BaseAddress}/api/");
}

async public Task<User> RegisterUser(PostUserDto userData)
{
    if (!CrossConnectivity.Current.IsConnected)
    {
        errorService.HandleError("Nie masz połączenia z internetem.");
        return null;
    }
    try
    {
        var jsonToSend = Newtonsoft.Json.JsonConvert.SerializeObject(userData);
        var content = new StringContent(jsonToSend, Encoding.UTF8, "application/json");
        var result = await client.PostAsync("users", content);
        if (result.StatusCode == System.Net.HttpStatusCode.OK)
        {
            var json = await result.Content.ReadAsStringAsync();
            Debug.WriteLine(json);
            var user = Newtonsoft.Json.JsonConvert.DeserializeObject<User>(json);

            var role = "BOOTH";
            if(user.Needy)
            {
                role = "SENIOR";
            }
            if(user.Helper)
            {
                role = "HELPER";
```

```
            }
            if(user.Helper && user.Needy)
            {
                role = "BOOTH";
            }
            CrossSecureStorage.Current.SetValue("UserRole", role);
            App.ROLE = role;

            return user;
        }
        else
        {
            errorService.HandleError("Wystąpiły problemy w aplikacji.");
            return null;
        }
    }
    catch(Exception ex)
    {
        Debug.WriteLine(ex.Message);
        errorService.HandleError("Wystąpiły problemy w aplikacji.");
        return null;
    }
}

async public Task<User> GetUserById(int id, string token)
{
    if (!CrossConnectivity.Current.IsConnected)
    {
        errorService.HandleError("Nie masz połączenia z internetem.");
        return null;
    }
    try
    {
        var result = await client.GetAsync<User>($"users/{id}", token);
```

```
            return result;
        }
        catch (UnauthorizedException)
        {
            var login = await authService.Login();
            if (login)
            {
                var result = await GetUserById(id, token);
                return result;
            }
            else
            {
                errorService.HandleError("Wystąpiły problemy w aplikacji.");
                return null;
            }
        }
        catch (Exception)
        {
            throw;
        }
    }

    async public Task<bool> UpdatePushNotificationToken(string token)
    {
        if (!CrossConnectivity.Current.IsConnected)
        {
            return false;
        }
        try
        {
            var result = await client.PutAsync($"users/fcm?fcmToken={token}", null, App.Token);
            Debug.WriteLine("UpdatePushNotificationToken "+result.StatusCode);
            if (result.StatusCode == System.Net.HttpStatusCode.OK)
            {
```

```
            return true;
        }
        else
        {
            return false;
        }
    }
    catch (UnauthorizedException)
    {
        var login = await authService.Login();
        if (login)
        {
            await UpdatePushNotificationToken(token);
            return true;
        }
        else
        {
            errorService.HandleError("Wystąpiły problemy w aplikacji.");
            return false;
        }
    }
    catch (Exception)
    {
        throw;
    }
}

async public Task<bool> UpdateUser(int id, User userData, string token)
{
    if (!CrossConnectivity.Current.IsConnected)
    {
        errorService.HandleError("Nie masz połączenia z internetem.");
        return false;
    }
```

```
try
{
    var userDataDto = new UpdateUserDto
    {
        Avatar = userData.Avatar,
        Name = userData.Name,
        Description = userData.Description,
        Gender = userData.Gender,
        Helper = userData.Helper,
        Needy = userData.Needy,
        NotificationActive = userData.NotificationsActive,
        NotificationRadius = userData.NotificationRadius,
        Phone = userData.Phone
    };

    var jsonToSend = Newtonsoft.Json.JsonConvert.SerializeObject(userDataDto);
    var stringContent = new StringContent(jsonToSend, Encoding.UTF8, "application/json");

    var result = await client.PutAsync($"users/{id}", stringContent, App.Token);
    Debug.WriteLine("UpdateUser " + result.StatusCode);
    if (result.StatusCode == System.Net.HttpStatusCode.OK)
    {
        return true;
    }
    else
    {
        return false;
    }
}
catch (UnauthorizedException)
{
    var login = await authService.Login();
    if (login)
    {
```

```
            var result = await UpdateUser(id, userData, token);
            return result;
        }
        else
        {
            errorService.HandleError("Wystąpiły problemy w aplikacji.");
            return false;
        }
    }
    catch (Exception)
    {
        throw;
    }
}

async public Task<bool> UpdateLocation(double longitude, double latitude)
{
    if (!CrossConnectivity.Current.IsConnected)
    {
        errorService.HandleError("Nie masz połączenia z internetem.");
        return false;
    }
    try
    {
        var dto = new Location
        {
            Longitude = longitude,
            Latitude = latitude
        };

        var jsonToSend = Newtonsoft.Json.JsonConvert.SerializeObject(dto);
        var stringContent = new StringContent(jsonToSend, Encoding.UTF8, "application/json");
```

```
            var result = await client.PutAsync($"users/{App.UserID}/location", stringContent, App.Token);
            Debug.WriteLine("UpdateLocation " + result.StatusCode);
            if (result.StatusCode == System.Net.HttpStatusCode.OK)
            {
                return true;
            }
            else
            {
                return false;
            }
        }
        catch (Exception)
        {
            throw;
        }
    }
  }
}
```

Styles:

CustomStyle:

```
using System;
using Xamarin.Forms;

namespace PoMOST.Styles
{
  public class CustomStyles
  {
    public static Color DarkBlue = Color.FromHex("#0eabfa");
    public static Color LightBlue = Color.FromHex("#56c3fd");
```

```
public static Style RegularText = new Style(typeof(Label))
{
   Setters =
   {
      new Setter { Property=Label.TextColorProperty, Value=DarkBlue},
      new Setter { Property=Label.FontSizeProperty, Value=16},
   }
};

public static Style WhiteButtonStyle = new Style(typeof(Button))
{
   Setters =
   {
      new Setter { Property=Button.TextColorProperty, Value=DarkBlue},
      new Setter { Property=Button.BorderColorProperty, Value=DarkBlue},
      new Setter { Property=Button.BorderWidthProperty, Value=1},
      new Setter { Property=Button.BackgroundColorProperty, Value=Color.White},
      new Setter { Property=Button.FontSizeProperty, Value=18},
   }
};

public static Style BlueButtonStyle = new Style(typeof(Button))
{
   Setters =
   {
      new Setter { Property=Button.TextColorProperty, Value=Color.White},
      new Setter { Property=Button.BackgroundColorProperty, Value=DarkBlue},
      new Setter { Property=Button.FontSizeProperty, Value=18},
   }
};

public static Style TitlePopupStyle = new Style(typeof(Button))
{
   Setters =
```

```
    {
        new Setter { Property=Label.TextColorProperty, Value=DarkBlue},
        new Setter { Property=Label.FontSizeProperty, Value=18}
    }
};

public static Style TextPopupStyle = new Style(typeof(Button))
{
    Setters =
    {
        new Setter { Property=Label.TextColorProperty, Value=DarkBlue },
        new Setter { Property=Label.FontSizeProperty, Value=18 }
    }
};

public static Style ProfileDarkTextMediumStyle = new Style(typeof(Button))
{
    Setters =
    {
        new Setter { Property=Label.TextColorProperty, Value = DarkBlue },
        new Setter { Property=Label.FontSizeProperty, Value = 18 }
    }
};

public static Style ProfileDarkTextRegularStyle = new Style(typeof(Button))
{
    Setters =
    {
        new Setter { Property=Label.TextColorProperty, Value = DarkBlue },
        new Setter { Property=Label.FontSizeProperty, Value = 18 }
    }
};

public static Style ProfileLightTextRegularStyle = new Style(typeof(Button))
```

```
        {
            Setters =
            {
                new Setter { Property=Label.TextColorProperty, Value = LightBlue },
                new Setter { Property=Label.FontSizeProperty, Value = 16 }
            }
        };

        public static Style ProfileTextBoxStyle = new Style(typeof(Button))
        {
            Setters =
            {
                new Setter { Property=Label.TextColorProperty, Value = DarkBlue },
                new Setter { Property=Label.FontSizeProperty, Value = 16 }
            }
        };

        public static Style RobotoLightTextStyle = new Style(typeof(Button))
        {
            Setters =
            {
                new Setter { Property=Label.TextColorProperty, Value = Color.White },
                new Setter { Property=Label.FontSizeProperty, Value = 36 }
            }
        };
    }
}
```

TwinTechs:

Controls:

PageViewContauner:

```
using System;
using Xamarin.Forms;

namespace TwinTechs.Controls
{
	//specialized class for showing a page within a page
	public class PageViewContainer : View
	{
		public PageViewContainer ()
		{
		}

		public static readonly BindableProperty ContentProperty = BindableProperty.Create<PageViewContainer,Page> (s => s.Content, null);

		public Page Content {
			get{ return (Page)GetValue (ContentProperty); }
			set{ SetValue (ContentProperty, value); }
		}
	}
}
```

ViewEffectExtensions:

```
using System;

namespace TwinTechs.Controls
{
	public enum ViewMaskerType
	{
		None = 0,
		Circle = 1,
		Triangle = 2,
		Square = 3
```

```
    }

    public interface IViewEffectExtensionProvider
    {
        void ApplyMaskToView (Xamarin.Forms.View view, ViewMaskerType maskType);

        void ToggleViewShadow (Xamarin.Forms.View view, bool isOn);
    }

    public static class ViewEffectExtensions
    {
        public static IViewEffectExtensionProvider ViewExtensionProvider { get; set; }

        public static void ApplyMask (this Xamarin.Forms.View view, ViewMaskerType maskType)
        {
            ViewExtensionProvider.ApplyMaskToView (view, maskType);
        }

        public static void ToggleShadow (this Xamarin.Forms.View view, bool isOn)
        {
            ViewExtensionProvider.ToggleViewShadow (view, isOn);
        }
    }
}
```

Extensions:

ViewExtensions:

```
using System;
using Xamarin.Forms;
```

```
namespace TwinTechs.Extensions
{
	public static class ViewExtensions
	{
		/// <summary>
		/// Gets the page to which an element belongs
		/// </summary>
		/// <returns>The page.</returns>
		/// <param name="element">Element.</param>
		public static Page GetParentPage (this VisualElement element)
		{
			if (element != null) {
				var parent = element.Parent;
				while (parent != null) {
					if (parent is Page) {
						return parent as Page;
					}
					parent = parent.Parent;
				}
			}
			return null;
		}
	}
}
```

ViewModels:

AddHelpRequestPageView:

```
using System;
using System.Diagnostics;
using PoMOST.Services.Navigation;
using PoMOST.Views.Popups;
using Xamarin.Forms;
```

```
namespace PoMOST.ViewModels
{

    public class AddHelpRequestPageViewModel
    {
        private INavigationService navigationService;

        public Command OpenPopupCommand { get; private set; }

        public AddHelpRequestPageViewModel(INavigationService navigationService)
        {
            this.navigationService = navigationService;

            OpenPopupCommand = new Command((obj) =>
            {
                Debug.WriteLine("click "+obj);
                OpenPopup(obj);
            });
        }

        async void OpenPopup(object obj)
        {
            switch (obj)
            {
                case 0:
                    Debug.WriteLine("Case 0");
                    await navigationService.ShowPopup(new ShopRequestPopup());
                    break;
                case 1:
                    Debug.WriteLine("Case 1");
                    await navigationService.ShowPopup(new HealthRequestPopup());
                    break;
                case 2:
                    Debug.WriteLine("Case 2");
```

```
                await navigationService.ShowPopup(new TransportRequestPopup());
                break;
            case 3:
                Debug.WriteLine("Case 3");
                await navigationService.ShowPopup(new PostRequestPopup());
                break;
            case 4:
                Debug.WriteLine("Case 4");
                await navigationService.ShowPopup(new AlarmRequestPopup());
                break;
            case 5:
                Debug.WriteLine("Case 5");
                await navigationService.ShowPopup(new InternetLearnRequestPopup());
                break;
            case 6:
                Debug.WriteLine("Case 6");
                await navigationService.ShowPopup(new OtherRequestPopup());
                break;
            default:
                Debug.WriteLine("default");
                break;
        }
    }
  }
}
```

AuthorsPageViewModel

```
using System;
using PoMOST.Services.Navigation;

namespace PoMOST.ViewModels
{
```

```
public class AuthorsPageViewModel : BasicModalPageViewModel
{
    public AuthorsPageViewModel(INavigationService navigationService) : base(navigationService)
    {
    }
}
}
```

BasicModalPageViewModel:

```
using System;
using PoMOST.Services.Navigation;
using Xamarin.Forms;

namespace PoMOST.ViewModels
{
    public class BasicModalPageViewModel
    {
        INavigationService navigationService;
        public Command CloseCommand;

        public BasicModalPageViewModel(INavigationService navigationService)
        {
            this.navigationService = navigationService;

            CloseCommand = new Command(async () =>
            {
                await navigationService.CloseModalAsync();
            });
        }
    }
}
```

BasicPageViewModel:

```
using System;
using System.ComponentModel;
using System.Runtime.CompilerServices;

namespace PoMOST.ViewModels
{
    public class BasicPageViewModel : INotifyPropertyChanged
    {
        bool isBusy;
        public bool IsBusy
        {
            get
            {
                return isBusy;
            }
            set
            {
                isBusy = value;
                OnPropertyChanged(nameof(IsBusy));
            }
        }

        public BasicPageViewModel()
        {
        }

        public event PropertyChangedEventHandler PropertyChanged;

        virtual public void OnAppear()
        {

        }
```

```
        virtual public void OnDisappear()
        {

        }

        protected virtual void OnPropertyChanged(string propertyName)
        {
            PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
        }
    }
}
```

CellsViewModels:

HelpRequestsPageViewModel:

```
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.ComponentModel;
using System.Diagnostics;
using System.Runtime.CompilerServices;
using System.Threading.Tasks;
using PoMOST.Models;
using PoMOST.Services;
using PoMOST.Services.Navigation;
using PoMOST.Views.Pages;
using PoMOST.Views.Popups;
using Xamarin.Forms;

namespace PoMOST.ViewModels
{
    public class HelpRequestsPageViewModel : BasicPageViewModel
```

```
{
    public Command ProfileCommand { get; private set; }
    public Command RefreshCommand { get; private set; }


    private INavigationService navigationService;
    private IHelpRequestsService helpRequestsService;

    public ObservableCollectionFast<HelpRequest> HelpRequests { get; private set; }
    public ObservableCollectionFast<HelpRequest> MyHelpRequests { get; private set; }

    bool IsInit = false;


    public HelpRequestsPageViewModel(INavigationService navigationService,
IHelpRequestsService helpRequestService)
    {
        this.navigationService = navigationService;
        this.helpRequestsService = helpRequestService;

        ProfileCommand = new Command(async () =>
        {
            await navigationService.OpenModalAsync<ProfilePage>();
        });

        RefreshCommand = new Command(async () =>
        {
            IsBusy = true;
            await GetData();
            IsBusy = false;
        });

        HelpRequests = this.helpRequestsService.Requests;
        MyHelpRequests = this.helpRequestsService.MyRequests;
```

```
}

async private Task Init()
{
    if (App.ROLE != "HELPER")
    {
        await GetMyRequests();
    }
    if (App.ROLE != "SENIOR")
    {
        await GetData();
    }
}

async private Task GetMyRequests()
{
    try
    {
        Debug.WriteLine("GetMyRequests");
        var result = await helpRequestsService.GetMyHelpRequests();
        Debug.WriteLine("GetMyRequests "+result);
    }
    catch (Exception exc)
    {
        Debug.WriteLine(exc.Message);
    }
}

async private Task GetData()
{
    try
    {
        var result = await helpRequestsService.GetHelpRequests(null);
```

```
            }
            catch (Exception exc)
            {
                Debug.WriteLine(exc.Message);
            }

        }

        async override public void OnAppear()
        {
            if (!IsInit)
            {
                await Init();
                IsInit = true;
            }
        }
    }
}
```

MainPageViewModel:

```
using System;
using System.Diagnostics;
using System.Threading.Tasks;
using Plugin.FirebasePushNotification;
using PoMOST.Services;
using PoMOST.Services.Navigation;
using Xamarin.Forms;

namespace PoMOST.ViewModels
{
    public class MainPageViewModel
    {
        INavigationService navigationService;
```

```
IAuthService authService;
IUserService userService;

public MainPageViewModel(INavigationService navigationService, IAuthService authService, IUserService userService)
{
    this.navigationService = navigationService;

    this.authService = authService;
    this.userService = userService;

    Init();
}

async Task Init()
{
    //await authService.Login();
    try
    {
        await userService.UpdatePushNotificationToken(CrossFirebasePushNotification.Current.Token);
    }
    catch (Exception e)
    {
        Debug.WriteLine(e.Message);
    }
}

public bool BackPageAsync()
{
    Debug.WriteLine(navigationService.GetStackSize());
    if (navigationService.GetStackSize() > 1)
    {
        navigationService.BackAsync();
        return true;
```

```
            }
            else
            {
                return false;
            }
        }


    }
}
```

NamePageViewModel:

```
using System;
using PoMOST.Services.Navigation;

namespace PoMOST.ViewModels
{
    public class NamePageViewModel : BasicModalPageViewModel
    {
        public NamePageViewModel(INavigationService navigationService) :
base(navigationService)
        {
        }
    }
}
```

PhoneNumberPageViewModel:

```
using System;
using PoMOST.Services.Navigation;

namespace PoMOST.ViewModels
{
```

```
    public class PhoneNumberPageViewModel : BasicModalPageViewModel
    {
        public PhoneNumberPageViewModel(INavigationService navigationService) :
base(navigationService)
        {
        }
    }
}
```

PopupsViewModels:

ProfilePageViewModel:

```
using System;
using System.Diagnostics;
using PoMOST.Services.Navigation;
using Xamarin.Forms;

namespace PoMOST.ViewModels
{
    public class ProfilePageViewModel : BasicModalPageViewModel
    {
        public Command OpenPhoneDialerCommand { get; private set; }
        public string Phone { get; set; }

        public ProfilePageViewModel(INavigationService navigationService):base(navigationService)
        {
            OpenPhoneDialerCommand = new Command((obj) =>
            {
                try
                {
                    Device.OpenUri(new Uri(string.Format("tel:{0}", Phone)));
                }
                catch (Exception exc)
```

```
            {
                Debug.WriteLine(exc.Message);
            }
        });
    }
  }
}
```

PregistrationViewModels:

SettingsPageViewModel:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Diagnostics;
using System.IO;
using System.Runtime.CompilerServices;
using System.Threading.Tasks;
using Plugin.Media;
using PoMOST.Models;
using PoMOST.Services;
using PoMOST.Services.Navigation;
using PoMOST.Views.Pages;
using PoMOST.Views.Popups;
using Xamarin.Forms;

namespace PoMOST.ViewModels
{
  public class SettingsPageViewModel : BasicPageViewModel
  {
    private INavigationService navigationService;
    IUserService userService;
    DisplayErrorService errorService;
```

```
IUploadService uploadService;

public Command OpenSettingsCommand { get; private set; }

public TapGestureRecognizer PhotoTap{ get; private set; }

private int textLimit = 500;

bool IsInit = false;

bool cleanOnDisapper = true;

User userData;

Stream currentStream;

private int textCounter;
public int TextCounter
{
    get
    {
        return textLimit - textCounter;
    }
    set
    {
        textCounter = value;
        OnPropertyChanged(nameof(TextCounter));
    }
}

string userName;
public string UserName
{
    get
```

```
    {
        return userName;
    }
    set
    {
        userName = value;
        OnPropertyChanged("UserName");
    }
}

string userPhone;
public string UserPhone
{
    get
    {
        return userPhone;
    }
    set
    {
        userPhone = value;
        OnPropertyChanged("UserPhone");
    }
}

string userDescription;
public string UserDescription
{
    get
    {
        Debug.WriteLine("description get");
        return userDescription;
    }
    set
    {
```

```
        Debug.WriteLine("description set");
        userDescription = value;
        OnPropertyChanged("UserDescription");
    }
}

int userDistance;
public int UserDistance
{
    get
    {
        return userDistance;
    }
    set
    {
        userDistance = value;
        SliderDistance = Ranges.IndexOf(userDistance);
        OnPropertyChanged("UserDistance");
    }
}

List<int> Ranges = new List<int>() { 100, 300, 500, 1000, 1500, 2000, 3000, 5000, 10000 };
int sliderDistance;
public int SliderDistance
{
    get
    {
        return sliderDistance;
    }
    set
    {
        sliderDistance = value;
        OnPropertyChanged("SliderDistance");
    }
```

```
}

bool userNotification;
public bool UserNotification
{
   get
   {
      return userNotification;
   }
   set
   {
      userNotification = value;
      OnPropertyChanged("UserNotification");
   }
}

string userAvatar;
public string UserAvatar
{
   get
   {
      if (userData != null)
      {
         if (string.IsNullOrEmpty(userAvatar))
         {
            Debug.WriteLine("TU DOTARLEM");
            var role = (userData.Needy) ? "senior" : "helper";
            return "avatar_" + role + "_" + userData.Gender;
         }
      }
      return userAvatar;
   }
   set
   {
```

```
            userAvatar = value;
            OnPropertyChanged("UserAvatar");
        }
    }

    ImageSource photoSource;
    public ImageSource PhotoSource
    {
        get
        {
            return photoSource;
        }
        set
        {
            Debug.WriteLine("TU DOTARLEM 2 "+value);
            photoSource = value;
            OnPropertyChanged("PhotoSource");
        }
    }



    public Command SaveAvatarCommand { get; private set; }
    public Command SaveDescriptionCommand { get; private set; }
    public Command SaveNotificationSettingsCommand { get; private set; }


    public SettingsPageViewModel(INavigationService navigationService, IUserService userService, DisplayErrorService errorService, IUploadService uploadService)
    {
        this.navigationService = navigationService;
        this.userService = userService;
        this.errorService = errorService;
        this.uploadService = uploadService;
```

```
OpenSettingsCommand = new Command((obj) =>
{
    Debug.WriteLine("click " + obj);
    OpenSettings(obj);
});

PhotoTap = new TapGestureRecognizer();
PhotoTap.Tapped += (s, e) => {
    Debug.WriteLine("Photo icon tap");
    OpenPhotoPopup();
};

SaveAvatarCommand = new Command(async () =>
{
    IsBusy = true;
    if(currentStream==null)
    {
        errorService.HandleError("Nie wybrano nowego zdjęcia");
        IsBusy = false;
        return;
    }

    var tempAvatar = userData.Avatar;
    try
    {
        byte[] buffer = new byte[currentStream.Length];
        currentStream.Read(buffer, 0, (int)currentStream.Length);
        var ms = new MemoryStream(buffer);
        var result = await uploadService.UploadFile(ms);

        if(result==null)
        {
            errorService.HandleError("Nie udało się wysłać nowego zdjącia");
            IsBusy = false;
```

```
            return;
        }

        userData.Avatar = result.ResourceHttpLink;
        var resultUpdate = await userService.UpdateUser(App.UserID, userData, App.Token);
        if(resultUpdate)
        {
            UserAvatar = result.ResourceHttpLink;
            errorService.HandleError("Avatar został zmieniony.", true);
        }
        else
        {
            errorService.HandleError("Nie udało się zapisać nowego zdjęcia");
            userData.Avatar = tempAvatar;
        }
    }
    catch(Exception exc)
    {
        Debug.WriteLine(exc.Message);
        errorService.HandleError("Nie udało się wysłać nowego zdjącia");

    }
});

SaveDescriptionCommand = new Command(async () =>
{
    IsBusy = true;
    var tempDescription = userData.Description;
    userData.Description = UserDescription;
    Debug.WriteLine(UserDescription);
    try
    {
        var result = await userService.UpdateUser(App.UserID, userData, App.Token);
        if(result)
```

```
        {
            errorService.HandleError("Opis został zmieniony.", true);
        }
        else
        {
            errorService.HandleError("Nie udało się zapisać nowego opisu");
            userData.Description = tempDescription;
        }
    }
    catch(Exception exc)
    {
        Debug.WriteLine(exc.Message);
    }
    IsBusy = false;
});

SaveNotificationSettingsCommand = new Command(async () =>
{
    IsBusy = true;
    var tempUserNotification = userData.NotificationsActive;
    var tempUserRadius = userData.NotificationRadius;

    userData.NotificationRadius = UserDistance;
    userData.NotificationsActive = UserNotification;

    try
    {
        var result = await userService.UpdateUser(App.UserID, userData, App.Token);
        if (result)
        {
            errorService.HandleError("Ustawienia zostały zmienione.", true);
        }
        else
        {
```

```
                errorService.HandleError("Nie udało się zapisać nowych ustawień");
                userData.NotificationRadius = tempUserRadius;
                userData.NotificationsActive = tempUserNotification;
            }
        }
        catch (Exception exc)
        {
            Debug.WriteLine(exc.Message);
        }
        IsBusy = false;
    });

}

async override public void OnAppear()
{
    if (!IsInit)
    {
        await Init();
        IsInit = true;
    }
}



public override void OnDisappear()
{
    if (!cleanOnDisapper)
        return;
    try
    {
        UserName = userData.Name;
        UserAvatar = userData.Avatar;
        UserPhone = userData.Phone;
```

```
            UserDistance = userData.NotificationRadius;
            UserDescription = userData.Description;
            UserNotification = userData.NotificationsActive;
            PhotoSource = UserAvatar;
        }
        catch(Exception e)
        {

        }
        if (currentStream != null)
            currentStream.Dispose();
    }

    async private Task Init()
    {

        try
        {
            var result = await userService.GetUserById(App.UserID, App.Token);
            userData = result;
            UserName = result.Name;
            UserPhone = result.Phone;
            UserDistance = result.NotificationRadius;
            UserNotification = result.NotificationsActive;
            UserDescription = result.Description;
            Debug.WriteLine(result.Description);
            UserAvatar = result.Avatar;
            PhotoSource = UserAvatar;

        }
        catch(Exception ex)
        {
            Debug.WriteLine(ex.Message);
        }
```

```
    }


    async Task OpenPhotoPopup()
    {
        cleanOnDisapper = false;
        var page = new PhotoPopup();
        await navigationService.ShowPopup(page);
        var result = await page.PopupClosedTask;
        if (string.IsNullOrEmpty(result))
            return;

        if(result == "TAKE")
        {
            await CrossMedia.Current.Initialize();
            if (!CrossMedia.Current.IsCameraAvailable || !CrossMedia.Current.IsTakePhotoSupported)
            {
                errorService.HandleError("Brak dostępnej kamery");
            }

            var file = await CrossMedia.Current.TakePhotoAsync(new Plugin.Media.Abstractions.StoreCameraMediaOptions()
            {
                Name = "face.jpg",
                PhotoSize = Plugin.Media.Abstractions.PhotoSize.Small,
                SaveToAlbum = false,
                DefaultCamera = Plugin.Media.Abstractions.CameraDevice.Front
            });

            if (file == null)
            {
                return;
            }
```

```
            currentStream = file.GetStreamWithImageRotatedForExternalStorage();

            PhotoSource = ImageSource.FromStream(() =>
            {
                var stream = file.GetStreamWithImageRotatedForExternalStorage();
                file.Dispose();
                return stream;
            });
        }
        else if(result == "PICK")
        {
            await CrossMedia.Current.Initialize();
            if (!CrossMedia.Current.IsPickPhotoSupported)
            {
                errorService.HandleError("Brak dostępnej galerii");
            }

            var file = await CrossMedia.Current.PickPhotoAsync(new
Plugin.Media.Abstractions.PickMediaOptions()
            {
                PhotoSize = Plugin.Media.Abstractions.PhotoSize.Small
            });

            if (file == null)
            {
                return;
            }

            currentStream = file.GetStreamWithImageRotatedForExternalStorage();

            PhotoSource = ImageSource.FromStream(() =>
            {
                var stream = file.GetStreamWithImageRotatedForExternalStorage();
                file.Dispose();
```

```
                return stream;
            });
        }
        cleanOnDisapper = true;
    }

    public bool ValidateName(string name)
    {
        if (string.IsNullOrWhiteSpace(name) || string.IsNullOrEmpty(name))
        {
            return false;
        }
        return true;
    }

    public bool ValidatePhone(string phone)
    {
        if (string.IsNullOrWhiteSpace(phone) || phone.Length < 9 || phone.Length > 9)
        {
            return false;
        }
        return true;
    }

    async void OpenSettings(object obj)
    {
        switch (obj)
        {
            case 0:
                Debug.WriteLine("Case 0");
                var namePage = await navigationService.OpenModalAsync<NamePage>(UserName)
as NamePage;
                var result = await namePage.ClosedTask;
                Debug.WriteLine("RESULT "+result);
```

```
            if(ValidateName(result))
            {
                var tempName = userData.Name;
                userData.Name = result;
                try
                {
                    var resultName = await userService.UpdateUser(App.UserID, userData, App.Token);
                    if (resultName)
                    {
                        UserName = userData.Name;
                        errorService.HandleError("Imię zostało zmienione.", true);
                    }
                    else
                    {
                        errorService.HandleError("Nie udało się zapisać nowego imienia.");
                        userData.Description = tempName;
                    }
                }
                catch (Exception exc)
                {
                    Debug.WriteLine(exc.Message);
                }
            }
            else
            {
                //errorService.HandleError("Niepoprawne imię. Nie może być puste.");
            }

            break;
        case 1:
            Debug.WriteLine("Case 1");
            var phonePage = await navigationService.OpenModalAsync<PhoneNumberPage>(UserPhone) as PhoneNumberPage;
            var resultFromPhone = await phonePage.ClosedTask;
```

```
                Debug.WriteLine("RESULT " + resultFromPhone);
                if (ValidatePhone(resultFromPhone))
                {
                    var tempPhone = userData.Phone;
                    userData.Phone = resultFromPhone;
                    try
                    {
                        var resultPhone = await userService.UpdateUser(App.UserID, userData, App.Token);
                        if (resultPhone)
                        {
                            UserPhone = userData.Phone;
                            errorService.HandleError("Numer telefonu został zmieniony.", true);
                        }
                        else
                        {
                            errorService.HandleError("Nie udało się zapisać nowego numeru telefonu.");
                            userData.Description = tempPhone;
                        }
                    }
                    catch (Exception exc)
                    {
                        Debug.WriteLine(exc.Message);
                    }
                }
                else
                {
                    //errorService.HandleError("Niepoprwny numer telefonu. Poprawny format to 9 cyfr.");
                }
                break;
            case 2:
                Debug.WriteLine("Case 2");
                try
                {
```

```
                Device.OpenUri(new
Uri("http://137.74.42.7:8088/uploads/docs/PoMOST_regulamin.pdf"));
            }
            catch(Exception e)
            {

            }
            //await navigationService.OpenModalAsync<TermsPage>();
            break;
        case 3:
            Debug.WriteLine("Case 3");
            await navigationService.OpenModalAsync<AuthorsPage>();
            break;
      }
    }
  }
}
```

TremsPageViewModel:

```
using System;
using PoMOST.Services.Navigation;

namespace PoMOST.ViewModels
{
  public class TermsPageViewModel : BasicModalPageViewModel
  {
    public TermsPageViewModel(INavigationService navigationService):base(navigationService)
    {
    }
  }
}
```

VoiceCommandsPageView:

```
using System;
using System.Diagnostics;
using PoMOST.Services;
using PoMOST.Services.Navigation;
using PoMOST.Views.Popups;
using Xamarin.Forms;

namespace PoMOST.ViewModels
{
   public class VoiceCommandsPageViewModel
   {
      public Command startCommand { get; private set; }
      ISpeechToText speechService;
      DisplayErrorService errorService;
      INavigationService navigationService;

      public VoiceCommandsPageViewModel(ISpeechToText speechService, DisplayErrorService errorService, INavigationService navigationService)
      {
         this.speechService = speechService;
         this.errorService = errorService;
         this.navigationService = navigationService;

         startCommand = new Command( () =>
         {
            try
            {
               speechService.StartRecognize();
            }
            catch(Exception)
            {
               errorService.HandleError("Aplikacja ma problem z uruchomieniem tej funkcji.");
            }
```

```
    });

    speechService.OnRecognize += SpeechService_OnRecognize;
}

void SpeechService_OnRecognize(object sender, string e)
{
    Debug.WriteLine(e);
    if(string.IsNullOrEmpty(e))
    {
        errorService.HandleError("Nie rozpoznano komendy.");
        return;
    }

    switch(e)
    {
        case "zakupy":
            navigationService.ShowPopup(new VoiceConfirmPopup("zakupy"));
            break;
        case "zdrowie":
            navigationService.ShowPopup(new VoiceConfirmPopup("zdrowie"));
            break;
        case "transport":
            navigationService.ShowPopup(new VoiceConfirmPopup("transport"));
            break;
        case "poczta":
            navigationService.ShowPopup(new VoiceConfirmPopup("poczta"));
            break;
        case "pomocy":
            navigationService.ShowPopup(new VoiceConfirmPopup("pomocy"));
            break;
        case "komputer":
            navigationService.ShowPopup(new VoiceConfirmPopup("komputer"));
            break;
```

```
            case "inne":
                navigationService.ShowPopup(new VoiceConfirmPopup("inne"));
                break;
            default:
                errorService.HandleError("Nie rozpoznano komendy.");
                break;
        }

    }
  }
}
```

Views:

Cells

HelpRequestCell

```
using System;
using System.Diagnostics;
using FFImageLoading.Svg.Forms;
using ImageCircle.Forms.Plugin.Abstractions;
using PoMOST.Helpers;
using PoMOST.Styles;
using PoMOST.ViewModels;
using PoMOST.ViewModels.CellsViewModels;
using PoMOST.Views.Controls;
using PoMOST.Views.Pages;
using Xamarin.Forms;

namespace PoMOST.Views.Cells
{
  public class HelpRequestCell : ViewCell
```

```
{
    public HelpRequestCell()
    {
        var vm = ContainerAccessor.Resolve<HelpRequestCellViewModel>();

        var roundedBox = new RoundedBoxView.Forms.Plugin.Abstractions.RoundedBoxView()
        {
            BackgroundColor = Color.White,
            CornerRadius = 5,
            BorderColor = CustomStyles.LightBlue,
            BorderThickness = 1
        };

roundedBox.SetBinding(RoundedBoxView.Forms.Plugin.Abstractions.RoundedBoxView.BorderC
olorProperty, "Category", converter:new BorderColorConverter());

        var grid = new Grid();

        grid.Children.Add(roundedBox,0,0);

        var mainLayout = new StackLayout()
        {
            Spacing = 20,
            Padding = new Thickness(20, 15)
        };
        grid.Children.Add(mainLayout);

        var avatar = new CircleImage
        {
            WidthRequest = 40,
            HeightRequest = 40,
            Aspect = Aspect.AspectFill
        };
        avatar.SetBinding(CircleImage.SourceProperty, "Author.AvatarSource");
```

```
        var nameLabel = new Label()
        {
            Style = CustomStyles.RegularText,
            FontSize = 18
        };
        nameLabel.SetBinding(Label.TextProperty, "Author.Name");

        var categoryLabel = new Label()
        {
            Style = CustomStyles.RegularText,
            TextColor = CustomStyles.LightBlue,
            FontSize = 16
        };
        categoryLabel.SetBinding(Label.TextProperty, "CategoryText");
        categoryLabel.SetBinding(Label.TextColorProperty, "Category", converter: new BorderColorConverter());

        var distanceLabel = new Label()
        {
            HorizontalOptions = LayoutOptions.EndAndExpand,
            VerticalOptions = LayoutOptions.Center,
            Style = CustomStyles.RegularText,
            TextColor = CustomStyles.LightBlue,
            FontSize = 14
        };
        distanceLabel.SetBinding(Label.TextProperty, "Distance", stringFormat:"{0}m");

        var arrowButton = new ContentView()
        {
            Padding = new Thickness(5,10),
            HorizontalOptions = LayoutOptions.End
        };
        arrowButton.Content = new SvgCachedImage { WidthRequest = 12, HeightRequest=8, Source = SvgImageSource.FromFile("ic_arrow_down.svg") };
```

```
var firstLayout = new StackLayout
{
   Orientation = StackOrientation.Horizontal,
   Spacing = 5,
   Children =
   {
      avatar,
      new StackLayout
      {
         Margin = new Thickness(15,0,0,0),
         Spacing = -5,
         VerticalOptions = LayoutOptions.Center,
         Children =
         {
            nameLabel,
            categoryLabel
         }
      },
      distanceLabel,
      arrowButton
   }
};

var callButton = new Button()
{
   Text = "ZADZWOŃ",
   Style = CustomStyles.BlueButtonStyle,
   HorizontalOptions = LayoutOptions.FillAndExpand
};
callButton.SetBinding(Button.CommandParameterProperty, ".");
callButton.Command = vm.OpenPhoneDialerCommand;

var profilButton = new Button()
```

```
        {
            Text = "PROFIL",
            Style = CustomStyles.WhiteButtonStyle,
            HorizontalOptions = LayoutOptions.FillAndExpand
        };
        profilButton.SetBinding(Button.CommandParameterProperty, ".");
        profilButton.Command = vm.OpenProfileCommand;


        var messageLabel = new Label
        {
            Style = CustomStyles.RegularText,
            TextColor = CustomStyles.LightBlue
        };
        messageLabel.SetBinding(Label.TextProperty, "Text");


        var buttonsGrid = new Grid();
        buttonsGrid.ColumnSpacing = 20;
        buttonsGrid.RowSpacing = 20;
        buttonsGrid.RowDefinitions.Add(new RowDefinition(){Height=new
GridLength(1,GridUnitType.Auto)});
        buttonsGrid.RowDefinitions.Add(new RowDefinition() { Height = new GridLength(1,
GridUnitType.Auto) });

        buttonsGrid.ColumnDefinitions.Add(new ColumnDefinition() { Width = new
GridLength(.5, GridUnitType.Star)});
        buttonsGrid.ColumnDefinitions.Add(new ColumnDefinition() { Width = new
GridLength(.5, GridUnitType.Star)});

        buttonsGrid.Children.Add(messageLabel,0,0);
        Grid.SetColumnSpan(messageLabel,2);

        buttonsGrid.Children.Add(profilButton, 0, 1);
        buttonsGrid.Children.Add(callButton, 1, 1);
```

```
/*var secondLayout = new StackLayout
{
    Orientation = StackOrientation.Horizontal,
    Children =
    {
        profilButton,
        callButton
    }
};*/
//secondLayout.IsVisible = false;

buttonsGrid.IsVisible = false;
var isOpen = false;
var tap = new TapGestureRecognizer();
tap.Tapped += async (sender, e) =>
{
    if (isOpen)
    {
        buttonsGrid.IsVisible = false;
        await arrowButton.RotateTo(0, 100);
        isOpen = false;
    }
    else
    {
        buttonsGrid.IsVisible = true;
        await arrowButton.RotateTo(180, 100);
        isOpen = true;
    }
};
arrowButton.GestureRecognizers.Add(tap);


mainLayout.Children.Add(firstLayout);
```

```
            mainLayout.Children.Add(buttonsGrid);

            grid.Margin = new Thickness(0, 0, 0, 5);

            View = grid;
        }
    }
}
```

MyHelpRequestCall

```
using System;
using System.Diagnostics;
using FFImageLoading.Svg.Forms;
using ImageCircle.Forms.Plugin.Abstractions;
using PoMOST.Models;
using PoMOST.Services;
using PoMOST.Styles;
using PoMOST.Views.Controls;
using Xamarin.Forms;

namespace PoMOST.Views.Cells
{
    public class MyHelpRequestCell : ContentView
    {
        public int Id { get; set; }

        public MyHelpRequestCell(HelpRequest requestData)
        {
            Id = requestData.Id;
            var roundedBox = new RoundedBoxView.Forms.Plugin.Abstractions.RoundedBoxView()
            {
                BackgroundColor = Color.White,
                CornerRadius = 5,
```

```
    BorderColor = CustomStyles.LightBlue,
    BorderThickness = 1
};

var grid = new Grid();

grid.Children.Add(roundedBox, 0, 0);

var mainLayout = new StackLayout()
{
    Spacing = 20,
    Padding = new Thickness(20, 15)
};
grid.Children.Add(mainLayout);

var avatar = new CircleImage
{
    Source = requestData.Author.AvatarSource,
    WidthRequest = 40,
    HeightRequest = 40,
    Aspect = Aspect.AspectFill
};

var nameLabel = new Label()
{
    Style = CustomStyles.RegularText,
    Text = "Ja",
    FontSize = 18
};

var categoryLabel = new Label()
{
    Style = CustomStyles.RegularText,
    TextColor = CustomStyles.LightBlue,
```

```
            Text = requestData.CategoryText,
            FontSize = 16
        };

        var distanceLabel = new Label()
        {
            HorizontalOptions = LayoutOptions.EndAndExpand,
            VerticalOptions = LayoutOptions.Center,
            Style = CustomStyles.RegularText,
            TextColor = CustomStyles.LightBlue,
            Text = requestData.CreationDate.ToString("dd.MM.yyyy"),
            FontSize = 14
        };

        var arrowButton = new ContentView()
        {
            Padding = new Thickness(5, 10),
            HorizontalOptions = LayoutOptions.End
        };
        arrowButton.Content = new SvgCachedImage { WidthRequest = 12, HeightRequest = 8,
Source = SvgImageSource.FromFile("ic_arrow_down.svg") };


        var firstLayout = new StackLayout
        {
            Orientation = StackOrientation.Horizontal,
            Spacing = 5,
            Children =
            {
                avatar,
                new StackLayout
                {
                    Margin = new Thickness(15,0,0,0),
                    Spacing = -5,
```

```
            VerticalOptions = LayoutOptions.Center,
            Children =
            {
                nameLabel,
                categoryLabel
            }
        },
        distanceLabel,
        arrowButton
    }
};

var helpRequestsService = ContainerAccessor.Resolve<IHelpRequestsService>();

var deleteButton = new Button()
{
    Text = "WYCOFAJ PROŚBĘ",
    Style = CustomStyles.WhiteButtonStyle,
    HorizontalOptions = LayoutOptions.FillAndExpand
};

deleteButton.Clicked += async (sender, e) =>
{
    try
    {
        await helpRequestsService.DeleteHelpRequest(requestData.Id);
    }
    catch (Exception ex)
    {
        Debug.WriteLine(ex.Message);
    }
};

var messageLabel = new Label
```

```
{
    Style = CustomStyles.RegularText,
    TextColor = CustomStyles.LightBlue
};
messageLabel.Text = requestData.Text;



var secondLayout = new StackLayout
{
    Spacing = 20,
    Children =
    {
        messageLabel,
        deleteButton
    }
};

secondLayout.IsVisible = false;
var isOpen = false;
var tap = new TapGestureRecognizer();
tap.Tapped += async (sender, e) =>
{
    if (isOpen)
    {
        secondLayout.IsVisible = false;
        await arrowButton.RotateTo(0, 100);
        isOpen = false;
    }
    else
    {
        secondLayout.IsVisible = true;
        await arrowButton.RotateTo(180, 100);
        isOpen = true;
```

```
            }
        };
        arrowButton.GestureRecognizers.Add(tap);


        mainLayout.Children.Add(firstLayout);
        mainLayout.Children.Add(secondLayout);

        //grid.Margin = new Thickness(0, 0, 0, 5);

        Content = grid;

    }
  }
}
```

Controls

AddHelpRequestButton:

```
using System;
using FFImageLoading.Svg.Forms;
using PoMOST.Styles;
using Xamarin.Forms;
using XamEffects;

namespace PoMOST.Views.Controls.AddHelpRequestButtonControl
{
  public class AddHelpRequestButton : ContentView
  {
    public int Id { get; set; }

    public Command ClickCommand { get; private set; }
```

```
public AddHelpRequestButton(string title, string activeIcon, int id, Command command)
{
    this.Id = id;
    ClickCommand = command;
    Grid grid = new Grid
    {
        Padding = new Thickness(0,0,20,0),
        InputTransparent = true,
        Children =
        {
            new Label { Text = title, FontSize = 20, TextColor = CustomStyles.DarkBlue, HorizontalTextAlignment = TextAlignment.Center, VerticalOptions = LayoutOptions.Center},
            new SvgCachedImage{WidthRequest=40, HeightRequest=40, Source = SvgImageSource.FromFile(activeIcon), HorizontalOptions = LayoutOptions.End, VerticalOptions = LayoutOptions.Center},
        }
    };

    HorizontalOptions = LayoutOptions.FillAndExpand;
    HeightRequest = 70;

    var text = new Label { VerticalOptions = LayoutOptions.Fill, HorizontalOptions = LayoutOptions.FillAndExpand,FontSize=20 };

    var layout = new Grid
    {
        Children =
        {
            new RoundedBoxView.Forms.Plugin.Abstractions.RoundedBoxView(){BackgroundColor = Color.White, CornerRadius=5, BorderColor = Color.LightBlue, BorderThickness = 1},
            text,
            grid
        }
    };
```

```
            Content = layout;

            TouchEffect.SetColor(text, Color.FromHex("#b5e5ff"));

            Commands.SetTap(text, new Command(() => ClickCommand?.Execute(Id)));
        }
    }
}
```

BorderEntryControl:

```
using System;
using System.Diagnostics;
using System.Threading.Tasks;
using PoMOST.Styles;
using Xamarin.Forms;

namespace PoMOST.Views.Controls
{
    public class BorderEntryControl : ContentView
    {
        public Editor entry { get; private set; }
        public string placeholder;

        public static readonly BindableProperty TextProperty = BindableProperty.Create("Text", typeof(string), typeof(BorderEntryControl), string.Empty, defaultBindingMode:BindingMode.TwoWay);

        public string Text
        {
            get
            {
                Debug.WriteLine("AER222W get "+TextProperty);
                return (string)GetValue(TextProperty);
            }
```

```
    set
    {
        Debug.WriteLine("AER222W "+value);
        SetValue(TextProperty, value);
    }
}

public string CurrentText
{
    get
    {
        if(entry.Text == placeholder)
        {
            return null;
        }
        return entry.Text;
    }
}

protected override void OnPropertyChanged(string propertyName = null)
{
    base.OnPropertyChanged(propertyName);
    Debug.WriteLine("AERW");
    if (propertyName == "Text")
    {
        if (!string.IsNullOrEmpty(Text))
        {
            entry.Text = Text;
        }
        else
        {

        }
    }
```

```
}

public BorderEntryControl(string placeholder)
{
    this.placeholder = placeholder;
    var grid = new Grid();

    Content = grid = new Grid
    {

    };

    grid.Children.Add(new RoundedBoxView.Forms.Plugin.Abstractions.RoundedBoxView()
    {
        BackgroundColor = Color.White,
        CornerRadius = 0,
        BorderColor = CustomStyles.DarkBlue,
        BorderThickness = 1
    });

    grid.Children.Add((entry = new Editor()
    {
        Style = CustomStyles.RegularText,
        Margin = 5
    }));

    entry.Text = placeholder;

    entry.Focused += async (object sender, FocusEventArgs e) =>
    {
        if(entry.Text == placeholder)
        {
            await Task.Delay(5);
            entry.Text = "";
```

```
            }
        };

        entry.Unfocused += (object sender, FocusEventArgs e) =>
        {
            if(string.IsNullOrWhiteSpace(entry.Text) || entry.Text == "")
            {
                entry.Text = placeholder;
            }
        };
    }
  }
}
```

BorderEntrySettingsControl:

```
using System;
using PoMOST.Styles;
using Xamarin.Forms;

namespace PoMOST.Views.Controls
{
  public class BorderEntrySettingsControl : ContentView
  {
    public Editor entry { get; private set; }
    string placeholder;

    public string Text
    {
        get
        {
            return entry.Text;
        }
    }
```

```
        public BorderEntrySettingsControl(string placeholder, Color borderColor)
        {
            this.placeholder = placeholder;
            var grid = new Grid();

            Content = grid = new Grid
            {

            };

            grid.Children.Add(new RoundedBoxView.Forms.Plugin.Abstractions.RoundedBoxView()
            {
                BackgroundColor = Color.White,
                CornerRadius = 0,
                BorderColor = borderColor,
                BorderThickness = 1
            });

            grid.Children.Add((entry = new Editor()
            {
                Style = CustomStyles.RegularText,
                Margin = 5
            }));

            entry.Text = placeholder;
        }
    }
}
```

DisplayErrorControl

```
using System;
using System.Diagnostics;
```

```
using System.Threading.Tasks;
using PoMOST.Models;
using PoMOST.Services;
using Xamarin.Forms;

namespace PoMOST.Views.Controls
{
   public class DisplayErrorControl :ContentView
   {
      Label errorLabel;
      BoxView background;
      public DisplayErrorControl()
      {
         InputTransparent = true;
         VerticalOptions = LayoutOptions.Start;
         Content = new Grid
         {
            Children =
            {
               (background = new BoxView() { Color = Color.Red, HeightRequest = 40 }),
               (errorLabel = new Label() { Text = "Nie masz połaczenia z internetem.", FontSize = 14, TextColor = Color.White, HorizontalTextAlignment = TextAlignment.Center, VerticalTextAlignment=TextAlignment.Center})
            }
         };
         this.TranslationY = -40;
         this.IsVisible = false;
      }

      async public Task Show(ErrorMsg errorMsg)
      {
         errorLabel.Text = errorMsg.Text;
         if(errorMsg.Success)
         {
            background.Color = Color.Green;
```

```
            }
            else
            {
                background.Color = Color.Red;
            }
            this.IsVisible = true;
            await this.TranslateTo(0,0,250);
            await Task.Delay(3000);
            await this.TranslateTo(0, -40, 250);
            this.IsVisible = false;
        }
    }
}
```

GadientBlueButton

```
using System;
using Xamarin.Forms;

namespace PoMOST.Views.Controls
{
    public class GradientBlueButton : GradientButton
    {
        public GradientBlueButton()
        {
            StartColor = Color.FromHex("#0eabfa");
            EndColor = Color.FromHex("#19ccd5");

            StartTouchColor = Color.FromHex("#19ccd5");
            EndTouchColor = Color.FromHex("#19ccd5");

            TextColor = Color.White;
            FontSize = 18;
        }
```

```
    }
}
```

GradientButton

```
using System;
using System.Windows.Input;
using Xamarin.Forms;

namespace PoMOST.Views.Controls
{
    public class GradientButton : Button
    {
        public Color StartColor { get; set; }
        public Color EndColor { get; set; }
        public Color StartTouchColor { get; set; }
        public Color EndTouchColor { get; set; }

        public GradientButton()
        {
            BackgroundColor = Color.Transparent;
        }
    }
}
```

Radio:

RadioButton:

```
using System;
using Xamarin.Forms;
using RoundedBoxView.Forms.Plugin.Abstractions;
using PoMOST.Styles;
```

```
namespace PoMOST.Views.Controls.Radio
{
    public class RadioButton : ContentView
    {
        public bool IsSelected { get; set; }

        RoundedBoxView.Forms.Plugin.Abstractions.RoundedBoxView border;
        RoundedBoxView.Forms.Plugin.Abstractions.RoundedBoxView circle;

        public RadioGroup parentControl;
        public int Id { get; private set; }

        public RadioButton(string title, int id)
        {
            this.Id = id;

            var button = new Grid()
            {
                HorizontalOptions = LayoutOptions.EndAndExpand
            };
            border = new RoundedBoxView.Forms.Plugin.Abstractions.RoundedBoxView()
            {
                WidthRequest = 20,
                HeightRequest = 20,
                CornerRadius = 20,
                BackgroundColor = Color.White,
                BorderThickness = 1,
                BorderColor = Color.FromHex("#55c2fd"),
                VerticalOptions = LayoutOptions.Center
            };
            circle = new RoundedBoxView.Forms.Plugin.Abstractions.RoundedBoxView()
            {
                BackgroundColor = Color.FromHex("#55c2fd"),
                WidthRequest = 10,
```

```
        HeightRequest =10,
        CornerRadius = 10,
        VerticalOptions = LayoutOptions.Center,
        HorizontalOptions = LayoutOptions.Center,
        IsVisible = false
    };

    button.Children.Add(border);
    button.Children.Add(circle);

    Content = new StackLayout
    {
        Orientation = StackOrientation.Horizontal,
        Children =
        {
            new Label{Text = title, Style=CustomStyles.RegularText},
            button
        }
    };

    var tapGesture = new TapGestureRecognizer();
    tapGesture.Tapped += (sender, e) =>
    {
        parentControl.Select(this);
    };

    button.GestureRecognizers.Add(tapGesture);

}

public void On()
{
    circle.IsVisible = true;
}
```

```
        public void Off()
        {
            circle.IsVisible = false;
        }
    }
}
```

RadioGroup:

```
using System;
using System.Collections.Generic;

namespace PoMOST.Views.Controls.Radio
{
    public class RadioGroup
    {
        List<RadioButton> buttons;
        public RadioButton CurrentSelected = null;

        public RadioGroup()
        {
            buttons = new List<RadioButton>();
        }

        public void AddRadioButton(RadioButton button)
        {
            button.parentControl = this;
            buttons.Add(button);
        }

        public void Select(RadioButton button)
        {
            if (CurrentSelected != null)
```

```
            {
                CurrentSelected.Off();
            }
            CurrentSelected = button;
            CurrentSelected.On();
        }

        public void Select(int radioId)
        {
            Select(buttons[radioId] as RadioButton);
        }
    }
}
```

RegistrationEntry:

```
using System;
using PoMOST.Styles;
using Xamarin.Forms;

namespace PoMOST.Views.Controls
{
    public class RegistrationEntry : ContentView
    {
        public Entry entry { get; private set; }
        string placeholder;

        bool isValidate;

        public string Text { get { return entry.Text; } }

        public RegistrationEntry(string placeholder, Keyboard keyboard)
        {
            this.placeholder = placeholder;
            var grid = new Grid();
```

```
Content = grid = new Grid
{
    HeightRequest = 50,
};

grid.Children.Add(new RoundedBoxView.Forms.Plugin.Abstractions.RoundedBoxView()
{
    BackgroundColor = Color.White,
    CornerRadius = 5
});

grid.Children.Add((entry = new Entry()
{
    Style = CustomStyles.RegularText,
    FontSize = 18,
    Margin = new Thickness(10,0,10,0),
    TextColor = Color.FromHex("#82d2fd"),
    PlaceholderColor = Color.FromHex("#82d2fd"),
    VerticalOptions = LayoutOptions.Center,
    Keyboard = keyboard
}));

entry.Placeholder = placeholder;

entry.TextChanged += (object sender, TextChangedEventArgs e) =>
{
    isValidate = true;
    if (isValidate)
    {
        isValidate = false;
        HideValidateLabel();
    }
};
```

```
        }
        public void ShowValidateLabel()
        {
            isValidate = true;
            entry.TextColor = Color.FromHex("#f3380c");
            entry.PlaceholderColor = Color.FromHex("#f3380c");
        }

        public void HideValidateLabel()
        {
            entry.TextColor = Color.FromHex("#82d2fd");
            entry.PlaceholderColor = Color.FromHex("#82d2fd");
        }
    }
}



//entry.TextChanged += (object sender, TextChangedEventArgs e) =>
//          {
//              if (isValidate)
//              {
//                  isValidate = false;
//                  HideValidateLabel();
//              }
//          };

//          Content = entry;
//      }

//      public void ShowValidateLabel()
//{
//    isValidate = true;
//    BackgroundColor = CustomColors.Red;
```

```
//}

//public void HideValidateLabel()
//{
//    BackgroundColor = CustomColors.Gray;
//}
   //}
```

RegistrationUserButton:

```
using System;
using System.Diagnostics;
using FFImageLoading.Svg.Forms;
using PoMOST.Services.Navigation;
using PoMOST.Styles;
using PoMOST.Views.Popups;
using Rg.Plugins.Popup.Pages;
using Xamarin.Forms;
using XamEffects;

namespace PoMOST.Views.Controls
{
   public class RegistrationUserButton<T> : ContentView where T: BasicNoBoxPopup
   {
      public int Id { get; set; }

      public string Value { get; set; }
      Label valueLabel;

      public RegistrationUserButton(string inputText, int id)
      {
         this.Id = id;
         Grid grid = new Grid
         {
```

```
            HeightRequest = 50,
            InputTransparent = true,
            Children =
            {
               new StackLayout
               {
                  Orientation = StackOrientation.Horizontal,
                  Spacing = 0,

                  Children =
                  {
                     (valueLabel = new Label { Text = inputText, Style = CustomStyles.ProfileDarkTextRegularStyle, TextColor = Color.FromHex("#82d2fd"), HorizontalOptions = LayoutOptions.StartAndExpand, VerticalOptions = LayoutOptions.Center, Margin = new Thickness(20,0,0,0)}),
                     new SvgCachedImage { Source = SvgImageSource.FromFile("ic_arrow_down"), VerticalOptions = LayoutOptions.Center, WidthRequest = 12, Margin = new Thickness(10,0,16,0) }
                  }
               }
            };

            HorizontalOptions = LayoutOptions.FillAndExpand;

            var text = new Label { VerticalOptions = LayoutOptions.Fill, HorizontalOptions = LayoutOptions.FillAndExpand };

            var layout = new Grid
            {
               Children =
               {
                  new RoundedBoxView.Forms.Plugin.Abstractions.RoundedBoxView(){ BackgroundColor = Color.White, CornerRadius = 5, BorderColor = Color.LightBlue, BorderThickness = 0 },
                  text,
                  grid
```

```
            }
        };

        Content = layout;

        TouchEffect.SetColor(text, Color.FromHex("#b5e5ff"));

        var navigationService = ContainerAccessor.Resolve<INavigationService>();

        var command = new Command(async () =>
        {
            var page = (T)Activator.CreateInstance(typeof(T));
            await navigationService.ShowPopup(page);
            var result = await page.PopupClosedTask;
            if(result!=null)
            {
                Debug.WriteLine("VALUE "+result);
                Value = result;
                valueLabel.TextColor = Color.FromHex("#82d2fd");
                valueLabel.Text = Value;

            }
        });

        Commands.SetTap(text, new Command(() => command.Execute(null)));
    }

    public void ShowValidateLabel()
    {
        valueLabel.TextColor = Color.FromHex("#f3380c");
    }
  }
}
```

SettingsAboutButton:

```
using System;
using FFImageLoading.Svg.Forms;
using PoMOST.Styles;
using Xamarin.Forms;
using XamEffects;

namespace PoMOST.Views.Controls
{
   public class SettingsAboutButton : ContentView
   {
      public int Id { get; set; }

      public Command ClickCommand { get; private set; }

      public SettingsAboutButton(string inputText, string arrowText, string activeIcon, int id, Command command)
      {
         this.Id = id;
         ClickCommand = command;
         Grid grid = new Grid
         {
            InputTransparent = true,
            Children =
            {
               new StackLayout
               {
                  HeightRequest = 70,
                  Orientation = StackOrientation.Horizontal,
                  Spacing = 0,

                  Children =
                  {
```

```
                new SvgCachedImage { Source = SvgImageSource.FromFile(activeIcon), VerticalOptions = LayoutOptions.Center, HeightRequest = 40, WidthRequest = 40, Margin = new Thickness(10,0,20,0) },
                new Label { Text = inputText, Style = CustomStyles.ProfileDarkTextRegularStyle, HorizontalOptions = LayoutOptions.StartAndExpand, VerticalOptions = LayoutOptions.Center},
                new StackLayout
                {
                    Spacing = 0,
                    Margin = new Thickness(0,0,10,16),
                    Orientation = StackOrientation.Horizontal,
                    HorizontalOptions = LayoutOptions.Fill,
                    VerticalOptions = LayoutOptions.End,
                    Children =
                    {
                        new Label { Text = arrowText, Style = CustomStyles.ProfileDarkTextRegularStyle, FontSize = 16 , VerticalOptions = LayoutOptions.Center, HorizontalOptions = LayoutOptions.End },
                        new SvgCachedImage { Source = SvgImageSource.FromFile("ic_arrow_down"), VerticalOptions = LayoutOptions.Center, HorizontalOptions = LayoutOptions.End, HeightRequest = 12, TranslationY = 3, Rotation = -90 }
                    }
                }
            }
        }
    };

    HorizontalOptions = LayoutOptions.FillAndExpand;
    HeightRequest = 70;

    var text = new Label { VerticalOptions = LayoutOptions.Fill, HorizontalOptions = LayoutOptions.FillAndExpand };

    var layout = new Grid
    {
        Children =
        {
```

```
            new RoundedBoxView.Forms.Plugin.Abstractions.RoundedBoxView(){ BackgroundColor = Color.White, CornerRadius = 5, BorderColor = Color.LightBlue, BorderThickness = 0 },
                text,
                grid
            }
        };

        Content = layout;

        TouchEffect.SetColor(text, Color.FromHex("#b5e5ff"));

        Commands.SetTap(text, new Command(() => ClickCommand?.Execute(Id)));
    }
  }
}
```

SettingPageButton:

```
using System;
using FFImageLoading.Svg.Forms;
using PoMOST.Styles;
using Xamarin.Forms;
using XamEffects;

namespace PoMOST.Views.Controls
{
  public class SettingsPageButton : ContentView
  {
    public static readonly BindableProperty TextProperty = BindableProperty.Create("Text", typeof(string), typeof(SettingsPageButton), string.Empty);

    public string Text
    {
      get
```

```
        {
            return (string)GetValue(TextProperty);
        }
        set
        {
            SetValue(TextProperty, value);
        }
    }

    protected override void OnPropertyChanged(string propertyName = null)
    {
        base.OnPropertyChanged(propertyName);
        if(propertyName== "Text")
        {
            textLabel.Text = Text;
        }
    }

    public Label textLabel;

    public int Id { get; set; }

    public Command ClickCommand { get; private set; }

    public SettingsPageButton(string title,string inputText,string arrowText, string activeIcon, int id, Command command)
    {
        this.Id = id;
        ClickCommand = command;
        Grid grid = new Grid
        {
            InputTransparent = true,
            Children =
            {
```

```
new StackLayout
{
    HeightRequest = 70,
    Orientation = StackOrientation.Horizontal,
    Spacing = 0,

    Children =
    {
        new SvgCachedImage { Source = SvgImageSource.FromFile(activeIcon), VerticalOptions = LayoutOptions.Center, HeightRequest = 40, WidthRequest = 40, Margin = new Thickness(10,0,20,0) },
        new StackLayout
        {
            HorizontalOptions = LayoutOptions.FillAndExpand,
            Spacing = 0,
            Padding = new Thickness(0,11,8,0),
            Children =
            {
                new Label { Text = title, Style = CustomStyles.ProfileLightTextRegularStyle, HorizontalOptions = LayoutOptions.Fill},
                new StackLayout
                {
                    Spacing = 0,
                    Orientation = StackOrientation.Horizontal,
                    HorizontalOptions = LayoutOptions.Fill,
                    Children =
                    {
                        (textLabel = new Label { Style = CustomStyles.ProfileDarkTextRegularStyle , HorizontalOptions = LayoutOptions.StartAndExpand, VerticalOptions = LayoutOptions.Center}),
                        new Label { Text = arrowText, Style = CustomStyles.ProfileDarkTextRegularStyle, FontSize = 16 , VerticalOptions = LayoutOptions.Center, HorizontalOptions = LayoutOptions.End },
                        new SvgCachedImage { Source = SvgImageSource.FromFile("ic_arrow_down"), VerticalOptions = LayoutOptions.Center, HorizontalOptions = LayoutOptions.End, HeightRequest = 12, Rotation = -90 }
                    }
```

```
                    }
                }
            }
        }
    }
};

HorizontalOptions = LayoutOptions.FillAndExpand;
HeightRequest = 70;

var text = new Label { VerticalOptions = LayoutOptions.Fill, HorizontalOptions = LayoutOptions.FillAndExpand };

var layout = new Grid
{
    Children =
    {
        new RoundedBoxView.Forms.Plugin.Abstractions.RoundedBoxView(){ BackgroundColor = Color.White, CornerRadius = 5, BorderColor = Color.LightBlue, BorderThickness = 0 },
        text,
        grid
    }
};

Content = layout;

TouchEffect.SetColor(text, Color.FromHex("#b5e5ff"));

Commands.SetTap(text, new Command(() => ClickCommand?.Execute(Id)));
    }
  }
}
```

Tabbar:

TabbarButton:

```
using System;
using System.Diagnostics;
using FFImageLoading.Svg.Forms;
using Xamarin.Forms;
using XamEffects;

namespace PoMOST.Views.Controls.Tabbar
{
   public class TabbarButton : ContentView
   {
      public TabbarControl parentControl;
      public int Id { get; set; }

      public TabbarButton(string title, string activeIcon, int id)
      {
         Id = id;
         HorizontalOptions = LayoutOptions.FillAndExpand;
         HeightRequest = 80;

         var text = new Label { VerticalOptions = LayoutOptions.Fill, HorizontalOptions = LayoutOptions.FillAndExpand };

         var layout = new Grid
         {
            Children =
            {
               text,
               new StackLayout
```

```
            {
                Padding = 10,
                InputTransparent= true,
                Children =
                {
                    new SvgCachedImage{WidthRequest=40, HeightRequest=40, Source = SvgImageSource.FromFile(activeIcon)},
                    new Label {Text = title, HorizontalTextAlignment = TextAlignment.Center, TextColor = Color.FromHex("0da9fd")}
                }
            }
        }

    };

    Content = layout;

    TouchEffect.SetColor(text, Color.FromHex("#b5e5ff"));

    Commands.SetTap(text, new Command(() => parentControl.Select(this)));

    Off();
  }

  public void On()
  {
    Opacity = 1;
  }

  public void Off()
  {
    Opacity = 0.5;
  }
 }
}
```

TabbarControl:

```
using System;
using Xamarin.Forms;

namespace PoMOST.Views.Controls.Tabbar
{
    public class TabbarControl : ContentView
    {
        StackLayout container;
        TabbarButton current;

        int counter = 0;

        public event EventHandler<int> Selected;

        public TabbarControl()
        {
            BackgroundColor = Color.White;

            Content = container = new StackLayout()
            {
                Spacing = 0,
                Orientation = StackOrientation.Horizontal
            };
        }

        public void AddTab(TabbarButton tab)
        {
            tab.parentControl = this;
            container.Children.Add(tab);
            counter++;
        }
```

```
        public void Select(TabbarButton tab)
        {
            if(current!=null)
            {
                current.Off();
            }
            current = tab;
            current.On();
            Selected?.Invoke(this, current.Id);
        }

        public void Select(int tabId)
        {
            Select(container.Children[tabId] as TabbarButton);

        }
    }
}
```

TouchContentView:

```
using System;
using Xamarin.Forms;

namespace PoMOST.Views.Controls
{
    public class TouchContentView :ContentView
    {
        public TouchContentView()
        {
        }
    }
}
```

Pages

AddHelpRequestPage:

```
using System;
using System.Diagnostics;
using PoMOST.Views.Controls;
using PoMOST.ViewModels;
using Xamarin.Forms;
using PoMOST.Views.Controls.AddHelpRequestButtonControl;

namespace PoMOST.Views.Pages
{
   public class AddHelpRequestPage : BasicPage
   {
      AddHelpRequestPageViewModel vm;

      public AddHelpRequestPage()
      {
         Title = "POTRZEBUJĘ";

         var scroll = new ScrollView();

         NavigationPage.SetHasNavigationBar(this, false);

         vm = ContainerAccessor.Resolve<AddHelpRequestPageViewModel>();

         var stackLayout = new StackLayout
         {
            Spacing = 10,
            Children =
            {
               new AddHelpRequestButton("Zakupy", "zakupy", 0, vm.OpenPopupCommand),
```

```
                new AddHelpRequestButton("Zdrowie", "zdrowie", 1, vm.OpenPopupCommand),
                new AddHelpRequestButton("Transport", "transport", 2, vm.OpenPopupCommand),
                new AddHelpRequestButton("Poczta", "poczta", 3, vm.OpenPopupCommand),
                new AddHelpRequestButton("Alarm", "alarm", 4, vm.OpenPopupCommand),
                new AddHelpRequestButton("Internet i komputer", "nauka_internetu", 5, vm.OpenPopupCommand),
                new AddHelpRequestButton("Inne", "inne", 6, vm.OpenPopupCommand)
            }
        };

        scroll.Content = stackLayout;

        PageContent.Content = scroll;
    }
  }
}
```

AuthorsPage:

```
using System;
using FFImageLoading.Svg.Forms;
using PoMOST.Styles;
using PoMOST.ViewModels;
using Xamarin.Forms;

namespace PoMOST.Views.Pages
{
  public class AuthorsPage : BasicModalPage
  {
    public AuthorsPage()
    {
      Title = "O AUTORACH";
      vm = ContainerAccessor.Resolve<AuthorsPageViewModel>();
```

```
        PageContent.Content = new ScrollView
        {
            Content = new StackLayout
            {
                Padding = new Thickness(10, 0, 10, 10),
                Children =
                {
                    new SvgCachedImage { Source = SvgImageSource.FromFile("autorzy"), VerticalOptions = LayoutOptions.Center, HeightRequest = 80, WidthRequest = 80 },
                    new Label { Style = CustomStyles.ProfileDarkTextMediumStyle, FontSize = 16, Text = "Dr Łukasz Krzyżowski ", Margin = new Thickness(0,15,0,0) },
                    new Label { Style = CustomStyles.ProfileDarkTextRegularStyle, FontSize = 16, Text = "Pracownik naukowy, analityk, data scientist i badacz. Od ponad dziesięciu lat zajmuje się realizacją badań UX w zakresie technologii wspierających opiekę nad osobami starszymi. Swoje doświadczenie zdobywał na licznych stażach badawczo-wdrożeniowych w Polsce i za granicą. Współpracownik MAXQDA w zakresie tworzenia nowych funkcjonalności oraz certyfikowany szkoleniowiec z komputerowej analizy danych.", Margin = new Thickness(0,10,0,0) },
                    new Label { Style = CustomStyles.ProfileDarkTextMediumStyle, FontSize = 16, Text = "Łukasz Malicki", Margin = new Thickness(0,25,0,0) },
                    new Label { Style = CustomStyles.ProfileDarkTextRegularStyle, FontSize = 16, Text = "Data scientist (w przeszłości: analityk biznesowy ds. IT i innowacji); prowadzi międzynarodowe projekty, których celem jest m.in. stworzenie nowoczesnych usług i rozwiązań dla seniorów, opiekunów jak i ośrodków służby zdrowia. Odpowiada za pakiety obejmujące analizowanie oczekiwań i potrzeb użytkowników, jak i projektowanie finalnych usług.", Margin = new Thickness(0,10,0,0) }
                }
            }
        };
    }
  }
}
```

BasicModalPage:

```
using System;
```

```
using System.Diagnostics;
using FFImageLoading.Svg.Forms;
using PoMOST.ViewModels;
using Xamarin.Forms;

namespace PoMOST.Views.Pages
{
    public class BasicModalPage : ContentPage
    {
        private Grid mainGrid;
        public ContentView PageContent;
        Label title;

        public BasicModalPageViewModel vm;

        public string Title
        {
            set
            {
                title.Text = value.ToUpper();
            }
        }

        public BasicModalPage()
        {
            NavigationPage.SetHasNavigationBar(this, false);
            BackgroundColor = Color.FromHex("#fafdff");

            mainGrid = new Grid
            {
                VerticalOptions = LayoutOptions.FillAndExpand
            };

            var closeButton = new ContentView()
```

```
        {
            Padding = new Thickness(5, 5),
            HorizontalOptions = LayoutOptions.End
        };
        closeButton.Content = new SvgCachedImage { WidthRequest = 18, HeightRequest = 18, Source = SvgImageSource.FromFile("ic_close.svg") };
        var closeTapGesture = new TapGestureRecognizer();
        closeTapGesture.Tapped += (sender, e) =>
        {
            vm?.CloseCommand.Execute(null);
        };
        closeButton.GestureRecognizers.Add(closeTapGesture);


        var topLayout = new Grid()
        {
            Padding = new Thickness(10,10,10,15),
            Children =
            {
                (title = new Label { HorizontalTextAlignment = TextAlignment.Center, TextColor = Color.White, VerticalOptions = LayoutOptions.Center, FontSize=18 }),
                closeButton
            }

        };



        mainGrid.Children.Add(new SvgCachedImage { Source = SvgImageSource.FromFile("bg_top.svg"), VerticalOptions = LayoutOptions.Start, TranslationY = -24 });
        mainGrid.Children.Add(
            new StackLayout
            {
                Children =
                {
                    topLayout,
```

```
                (PageContent = new ContentView())
            }

        }
      );


      Content = mainGrid;
    }
  }
}
```

HelpRequestsPage:

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using FFImageLoading.Svg.Forms;
using PoMOST.Models;
using PoMOST.Styles;
using PoMOST.ViewModels;
using PoMOST.Views.Cells;
using PoMOST.Views.Controls.Radio;
using Xamarin.Forms;

namespace PoMOST.Views.Pages
{
  public class HelpRequestsPage : BasicPage
  {
    HelpRequestsPageViewModel vm;
    ListView list;

    List<MyHelpRequestCell> myRequestItems;
```

```
public HelpRequestsPage()
{
    NavigationPage.SetHasNavigationBar(this, false);

    Title = "Pomogę";

    BindingContext = vm = ContainerAccessor.Resolve<HelpRequestsPageViewModel>();

    myRequestItems = new List<MyHelpRequestCell>();

    var myRequestContent = new StackLayout();
    var headerLayout = new StackLayout()
    {

    };

    var emptyMyList = new Grid
    {
        IsVisible = true,
        Children =
        {
            new RoundedBoxView.Forms.Plugin.Abstractions.RoundedBoxView
            {
                BackgroundColor = Color.White,
                CornerRadius = 5,
                BorderColor = CustomStyles.LightBlue,
                BorderThickness = 1,
                HorizontalOptions = LayoutOptions.Fill,
                VerticalOptions = LayoutOptions.Fill
            },
            new StackLayout
            {
                Padding = new Thickness(24,15),
```

```
                Spacing = 10,
                Orientation = StackOrientation.Horizontal,
                Children =
                {
                  new SvgCachedImage{ WidthRequest=40, HeightRequest=40, Source = SvgImageSource.FromFile("brak.svg"), VerticalOptions = LayoutOptions.Center, HorizontalOptions = LayoutOptions.Start },
                  new Label
                  {
                    Style = CustomStyles.RegularText,
                    VerticalOptions = LayoutOptions.Center,
                    HorizontalOptions = LayoutOptions.FillAndExpand,
                    Text = "Nie ma obecnie żadnych Twoich próśb. Jeżeli czegoś potrzebujesz dodaj prośbę. :)"
                  }
                }
              }
            }
        };

        var emptyRequestsList = new Grid
        {
          IsVisible = true,
          Children =
          {
            new RoundedBoxView.Forms.Plugin.Abstractions.RoundedBoxView
            {
              BackgroundColor = Color.White,
              CornerRadius = 5,
              BorderColor = CustomStyles.LightBlue,
              BorderThickness = 1,
              HorizontalOptions = LayoutOptions.Fill,
              VerticalOptions = LayoutOptions.Fill
            },
            new StackLayout
```

```
            {
                Padding = new Thickness(24,15),
                Spacing = 10,
                Orientation = StackOrientation.Horizontal,
                Children =
                {
                    new SvgCachedImage{ WidthRequest=40, HeightRequest=40, Source = SvgImageSource.FromFile("brak.svg"), VerticalOptions = LayoutOptions.Center, HorizontalOptions = LayoutOptions.Start },
                    new Label
                    {
                        Style = CustomStyles.RegularText,
                        VerticalOptions = LayoutOptions.Center,
                        HorizontalOptions = LayoutOptions.FillAndExpand,
                        Text = "Nie ma obecnie żadnych nowych próśb. Ale to super, że chcesz pomóc! :)"
                    }
                }
            }
        };

        list = new ListView()
        {
            HasUnevenRows = true,
            SeparatorVisibility = SeparatorVisibility.None,
            ItemTemplate = new DataTemplate(typeof(HelpRequestCell)),
            VerticalOptions = LayoutOptions.Fill,
            Header = headerLayout
        };

        list.IsPullToRefreshEnabled = true;
        list.RefreshCommand = vm.RefreshCommand;
        list.SetBinding(ListView.IsRefreshingProperty, "IsBusy");
```

```
        PageContent.Content = list;
        if (App.ROLE != "SENIOR")
        {
            list.ItemsSource = vm.HelpRequests;
        }

        if (App.ROLE != "HELPER")
        {
            Device.BeginInvokeOnMainThread(() =>
            {
                vm.MyHelpRequests.CollectionChanged += (object sender,
System.Collections.Specialized.NotifyCollectionChangedEventArgs e) =>
                {
                    Debug.WriteLine("CollectionChanged " + (e.NewItems == null));
                    Debug.WriteLine("CollectionChanged " + (e.OldItems == null));
                    if (e.NewItems == null && e.OldItems == null)
                    {
                        for (var i = 0; i < vm.MyHelpRequests.Count(); i++)
                        {
                            var item = new MyHelpRequestCell(vm.MyHelpRequests[i] as HelpRequest);
                            myRequestContent.Children.Add(item);
                            myRequestItems.Add(item);
                        }

                    }

                    if (e.NewItems != null && e.NewItems.Count > 0)
                    {

                        for (var i = 0; i < e.NewItems.Count; i++)
                        {
                            Debug.WriteLine((e.NewItems[i] as HelpRequest));
                            var item = new MyHelpRequestCell(e.NewItems[i] as HelpRequest);
                            myRequestContent.Children.Add(item);
```

```
                    myRequestItems.Add(item);
                }
            }
            if (e.OldItems != null && e.OldItems.Count > 0)
            {
                for (var i = 0; i < e.OldItems.Count; i++)
                {
                    var item = myRequestItems.Where(o => o.Id == (e.OldItems[i] as HelpRequest).Id).FirstOrDefault();
                    if (item != null)
                    {
                        myRequestContent.Children.Remove(item);
                        myRequestItems.Remove(item);
                    }
                }
            }

            Debug.WriteLine("testttttttt");
            if (vm.MyHelpRequests.Count == 0)
            {
                emptyMyList.IsVisible = true;
            }
            else
            {
                emptyMyList.IsVisible = false;
            }
        };
    });
}
else
{
    emptyMyList.IsVisible = false;
}
```

```
        vm.HelpRequests.CollectionChanged += (object sender,
System.Collections.Specialized.NotifyCollectionChangedEventArgs e) =>
        {

            if (vm.HelpRequests.Count == 0)
            {
                emptyRequestsList.IsVisible = true;
            }
            else
            {
                emptyRequestsList.IsVisible = false;
            }
        };

        if (App.ROLE != "HELPER")
        {
            headerLayout.Children.Add(new Label { Text = "MOJE PROŚBY", Style =
CustomStyles.RegularText, TextColor = Color.White });
            headerLayout.Children.Add(emptyMyList);
            headerLayout.Children.Add(myRequestContent);
            if (App.ROLE != "SENIOR")
            {
                headerLayout.Children.Add(new Label { Text = "PROŚBY INNYCH", Style =
CustomStyles.RegularText, Margin = new Thickness(0, 15, 0, 5) });
            }
        }
        if (App.ROLE != "SENIOR")
        {
            headerLayout.Children.Add(emptyRequestsList);
        }
    }

    protected override void OnAppearing()
    {
        base.OnAppearing();
```

```
            vm.OnAppear();
        }

    }
}
```

ICustomTabbedPage:

```
using System;
using Xamarin.Forms;

namespace PoMOST.Views.Pages
{
    public interface ICustomTabbedPage
    {
        Page CurrentPage { get; }
    }
}
```

MainPage:

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Threading.Tasks;
using FFImageLoading.Svg.Forms;
using Plugin.FirebasePushNotification;
using PoMOST.Models;
using PoMOST.Services;
using PoMOST.ViewModels;
using PoMOST.Views.Controls;
using PoMOST.Views.Controls.Tabbar;
using TwinTechs.Controls;
```

```
using Xamarin.Forms;

namespace PoMOST.Views.Pages
{
   public class MainPage : ContentPage, ICustomTabbedPage
   {
      MainPageViewModel vm;

      private PageViewContainer pagesContainer;

      List<Page> mainPages;
      private TabbarControl tabbar;

      bool isInit = false;
      bool isSelected = false;

      DisplayErrorControl errorControl;
      DisplayErrorService errorService;

      public MainPage()
      {
         BackgroundColor = Color.FromHex("#fafdff");

         vm = ContainerAccessor.Resolve<MainPageViewModel>();
         errorService = ContainerAccessor.Resolve<DisplayErrorService>();

         pagesContainer = new TwinTechs.Controls.PageViewContainer() { VerticalOptions = LayoutOptions.FillAndExpand };
         //pagesContainer.BackgroundColor = Color.Transparent;

         tabbar = new TabbarControl();
         tabbar.AddTab(new TabbarButton("Pomogę", "needhelp_ic_active.svg",0));
         if (App.ROLE != "HELPER")
         {
```

```
            tabbar.AddTab(new TabbarButton("Potrzebuję", "help_ic_active.svg",1));
            tabbar.AddTab(new TabbarButton("Mówię", "speak_ic_active.svg",2));
        }
        tabbar.AddTab(new TabbarButton("Ustawienia", "settings_ic_active.svg",3));

        errorControl = new DisplayErrorControl();

        Content = new Grid
        {
            VerticalOptions = LayoutOptions.FillAndExpand,
            Children =
            {
                new SvgCachedImage { Source = SvgImageSource.FromFile("bg_top.svg"), VerticalOptions = LayoutOptions.Start, TranslationY = -24 },
                new StackLayout
                {
                    Spacing = 0,
                    Children =
                    {
                        pagesContainer,
                        tabbar
                    }
                },
                errorControl
            }
        };

        mainPages = new List<Page>();
        mainPages.Add(new NavigationPage(new HelpRequestsPage()));
        mainPages.Add(new NavigationPage(new AddHelpRequestPage()));
        mainPages.Add(new NavigationPage(new VoiceCommandsPage()));
        mainPages.Add(new NavigationPage(new SettingsPage()));

        tabbar.Selected += (object sender, int e) =>
```

```
    {
        if (isSelected)
            return;
        isSelected = true;
        Page pageToOpen = mainPages[e];

        if (pageToOpen != null)
        {
            pagesContainer.Content = pageToOpen;
        }
        else
        {
            throw new NullReferenceException();
        }
        isSelected = false;
    };

    CrossFirebasePushNotification.Current.OnNotificationReceived += (s, p) =>
    {
        Debug.WriteLine("GOT MESSAGE");
        Debug.WriteLine(Newtonsoft.Json.JsonConvert.SerializeObject(p.Data));
    };
    errorService.OnError += ErrorService_OnError;
}

protected override void OnAppearing()
{
    base.OnAppearing();
    Debug.WriteLine("OnAPPEAR main");

    if(!isInit)
    {
        tabbar.Select(0);
        isInit = true;
```

```
        }

    }

    protected override void OnDisappearing()
    {
        base.OnDisappearing();

    }

    async void ErrorService_OnError(object sender, ErrorMsg e)
    {
        await errorControl.Show(e);
    }

    public Page CurrentPage
    {
        get
        {
            return pagesContainer.Content;
        }
    }

    protected override bool OnBackButtonPressed()
    {
        var result = vm.BackPageAsync();
        return result;
    }
  }
}
```

NamePage:

```
using System;
```

```
using System.Threading.Tasks;
using FFImageLoading.Svg.Forms;
using PoMOST.Services.Navigation;
using PoMOST.Styles;
using PoMOST.ViewModels;
using PoMOST.Views.Controls;
using Xamarin.Forms;

namespace PoMOST.Views.Pages
{
    public class NamePage : BasicModalPage
    {
        public Task<string> ClosedTask => tcs.Task;
        public string Value { get; set; }
        private TaskCompletionSource<string> tcs { get; set; }

        BorderEntrySettingsControl nameLabel;

        public NamePage(string userName)
        {
            Title = "POPRAWA IMIENIA";
            vm = ContainerAccessor.Resolve<NamePageViewModel>();

            var navigationService = ContainerAccessor.Resolve<INavigationService>();

            tcs = new TaskCompletionSource<string>();

            var cancelCommand = new Command(async () =>
            {
                Value = null;
                await navigationService.CloseModalAsync();
            });

            var saveCommand = new Command(async () =>
```

```
        {
            Value = nameLabel.Text;
            await navigationService.CloseModalAsync();
        });

        PageContent.Content = new StackLayout
        {
            Padding = new Thickness(10, 0, 10, 10),
            Children =
            {
                new SvgCachedImage { Source = SvgImageSource.FromFile("profil"), VerticalOptions = LayoutOptions.Center, HeightRequest = 80, WidthRequest = 80},
                new Label { Style = CustomStyles.ProfileDarkTextMediumStyle, FontSize = 16, Text = "Wprowadź poprawione imię", Margin = new Thickness(0,15,0,0) },
                (nameLabel = new BorderEntrySettingsControl(userName,Color.FromHex("#d0eeff")){ Margin = new Thickness(0, 10, 0, 0) }),
                new GradientBlueButton { Text = "ZMIEŃ", Margin = new Thickness(0,19,0,0), Command = saveCommand},
                new Button{ Text = "ANULUJ", Style = CustomStyles.WhiteButtonStyle ,HorizontalOptions = LayoutOptions.FillAndExpand, Margin = new Thickness(0,10,0,0), Command=cancelCommand}
            }
        };
    }

    protected override void OnDisappearing()
    {
        base.OnDisappearing();
        tcs.SetResult(Value);

    }
  }
}
```

PhoneNumberPage:

```
using System;
using System.Threading.Tasks;
using FFImageLoading.Svg.Forms;
using PoMOST.Services.Navigation;
using PoMOST.Styles;
using PoMOST.ViewModels;
using PoMOST.Views.Controls;
using Xamarin.Forms;

namespace PoMOST.Views.Pages
{
   public class PhoneNumberPage : BasicModalPage
   {
      public Task<string> ClosedTask => tcs.Task;
      public string Value { get; set; }
      private TaskCompletionSource<string> tcs { get; set; }

      BorderEntrySettingsControl phoneLabel;

      public PhoneNumberPage(string userPhone)
      {
         Title = "POPRAWA IMIENIA";
         vm = ContainerAccessor.Resolve<PhoneNumberPageViewModel>();

         var navigationService = ContainerAccessor.Resolve<INavigationService>();

         tcs = new TaskCompletionSource<string>();

         var cancelCommand = new Command(async () =>
         {
            Value = null;
            await navigationService.CloseModalAsync();
         });
```

```
        var saveCommand = new Command(async () =>
        {
            Value = phoneLabel.Text;
            await navigationService.CloseModalAsync();
        });

        PageContent.Content = new StackLayout
        {
            Padding = new Thickness(10, 0, 10, 10),
            Children =
            {
                new SvgCachedImage { Source = SvgImageSource.FromFile("telefon"), VerticalOptions = LayoutOptions.Center, HeightRequest = 80, WidthRequest = 80},
                new Label { Style = CustomStyles.ProfileDarkTextMediumStyle, FontSize = 16, Text = "Wprowadź nowy numer telefonu", Margin = new Thickness(0,15,0,0) },
                (phoneLabel = new BorderEntrySettingsControl(userPhone,Color.FromHex("#d0eeff")){ Margin = new Thickness(0, 10, 0, 0) }),
                new GradientBlueButton { Text = "ZMIEŃ", Margin = new Thickness(0,19,0,0), Command = saveCommand},
                new Button{ Text = "ANULUJ", Style = CustomStyles.WhiteButtonStyle ,HorizontalOptions = LayoutOptions.FillAndExpand, Margin = new Thickness(0,10,0,0), Command=cancelCommand },
            }
        };
    }

    protected override void OnDisappearing()
    {
        base.OnDisappearing();
        tcs.SetResult(Value);

    }
  }
}
```

ProfilePage:

```
using System;
using FFImageLoading.Svg.Forms;
using ImageCircle.Forms.Plugin.Abstractions;
using PoMOST.Models;
using PoMOST.Styles;
using PoMOST.ViewModels;
using PoMOST.Views.Controls;
using Xamarin.Forms;

namespace PoMOST.Views.Pages
{
   public class ProfilePage : BasicModalPage
   {

      public ProfilePage(User userData)
      {
         Title = "PROFIL";
         vm = ContainerAccessor.Resolve<ProfilePageViewModel>();
         (vm as ProfilePageViewModel).Phone = userData.Phone;

         var roundedDataBox = new
RoundedBoxView.Forms.Plugin.Abstractions.RoundedBoxView
         {
            BackgroundColor = Color.White,
            CornerRadius = 5,
            BorderColor = Color.LightBlue,
            BorderThickness = 1,
            VerticalOptions = LayoutOptions.FillAndExpand
         };

         var roundedTextBox = new
RoundedBoxView.Forms.Plugin.Abstractions.RoundedBoxView
```

```
{
    BackgroundColor = Color.White,
    CornerRadius = 0,
    BorderColor = Color.LightBlue,
    BorderThickness = 1
};



var dataGrid = new Grid
{
    VerticalOptions = LayoutOptions.FillAndExpand
};

var textGrid = new Grid
{
    Margin = new Thickness(0,20,0,0)
};

dataGrid.Children.Add(roundedDataBox, 0, 0);

textGrid.Children.Add(roundedTextBox, 0, 0);

var lableDateLayout = new StackLayout()
{
    HeightRequest = 140,
    //BackgroundColor = Color.White,
    Spacing = 0,
    Children =
    {
        new StackLayout
        {
            HeightRequest = 70,
            Orientation = StackOrientation.Horizontal,
```

```
            Spacing = 0,


            Children =
            {
              new SvgCachedImage { Source = SvgImageSource.FromFile("profil"), VerticalOptions = LayoutOptions.Center, HeightRequest = 40, WidthRequest = 40, Margin = new Thickness(10,15,20,0) },
              new StackLayout
              {
                Spacing = 0,
                Padding = new Thickness(0,11,0,0),
                Children =
                {
                  new Label { Text = "Imię" , Style = CustomStyles.ProfileLightTextRegularStyle },
                  new Label { Text = userData.Name , Style = CustomStyles.ProfileDarkTextRegularStyle }
                }
              }
            },
            new BoxView { Margin = new Thickness(0, 15, 0, 0), HeightRequest = 2, BackgroundColor = Color.FromHex("#330da9fd")},
            new StackLayout
            {
              HeightRequest = 70,
              Orientation = StackOrientation.Horizontal,
              Spacing =0,


              Children =
              {
                new SvgCachedImage { Source = SvgImageSource.FromFile("telefon"), VerticalOptions = LayoutOptions.CenterAndExpand, HeightRequest = 40, WidthRequest = 40 , Margin = new Thickness(10,13,20,15) },
                new StackLayout
                {
```

```
                    Spacing = 0,
                    Padding = new Thickness(0,11,0,13),
                    Children =
                    {
                        new Label { Text = "Telefon", Style = CustomStyles.ProfileLightTextRegularStyle },
                        new Label { Text = userData.Phone, Style = CustomStyles.ProfileDarkTextRegularStyle}
                    }
                }
            }
        }
    };
    dataGrid.Children.Add(lableDateLayout);

    var lableTextLayout = new Label
    {
        Text = userData.Description,
        Style = CustomStyles.ProfileTextBoxStyle,
        HeightRequest = 139,
        Margin = new Thickness(21, 15, 24, 0),

    };
    textGrid.Children.Add(lableTextLayout);

    var mainScroll = new ScrollView
    {
        Content=new Grid
        {
            VerticalOptions = LayoutOptions.FillAndExpand,
            Children =
            {
                new StackLayout
                {
```

```
                Padding = new Thickness(10,0),
                Spacing = 0,
                Children =
                {
                    new CircleImage {
                        FillColor = Color.White,
                        BorderColor = Color.FromHex("#19ccd5"),
                        BorderThickness = 10,
                        HeightRequest = 140,
                        WidthRequest = 140,
                        Aspect = Aspect.AspectFill,
                        HorizontalOptions = LayoutOptions.Center,
                        Source = userData.AvatarSource
                    },
                    new Label { Text = "Dane kontaktowe", Style = CustomStyles.
ProfileDarkTextMediumStyle, Margin = 10 },
                    dataGrid,
                    textGrid,
                    new GradientBlueButton { Text = "ZADZWOŃ", VerticalOptions =
LayoutOptions.EndAndExpand, Margin = new Thickness(0,10,0,16), Command = (vm as
ProfilePageViewModel).OpenPhoneDialerCommand}
                }
            }
        }
    };

    PageContent.Content = mainScroll;
  }
 }
}
```

Registration:

GeolocationPage:

```
using System;
using ImageCircle.Forms.Plugin.Abstractions;
using PoMOST.Helpers;
using PoMOST.Models;
using PoMOST.Services;
using PoMOST.Styles;
using PoMOST.ViewModels.RegistrationViewModels;
using PoMOST.Views.Controls;
using Xamarin.Forms;
using XFGloss;

namespace PoMOST.Views.Pages.Registration
{
    public class GeolocationPage : ContentPage
    {
        GeolocationPageViewModel vm;
        DisplayErrorControl errorControl;
        DisplayErrorService errorService;
        Button acceptButton;
        ActivityIndicator loader;

        public GeolocationPage(PostUserDto userData)
        {
            NavigationPage.SetHasNavigationBar(this, false);

            errorService = ContainerAccessor.Resolve<DisplayErrorService>();

            BindingContext = vm = ContainerAccessor.Resolve<GeolocationPageViewModel>();
            vm.UserData = userData;

            var bkgrndGradient = new Gradient()
            {
                Rotation = 0,
```

```
            Steps = new GradientStepCollection()
            {
                new GradientStep(Color.FromHex("#19ccd5"), 0),
                new GradientStep(Color.FromHex("#0eabfa"), 1)
            }
        };

        var text = new Label { VerticalOptions = LayoutOptions.Fill, HorizontalOptions = LayoutOptions.FillAndExpand };

        ContentPageGloss.SetBackgroundGradient(this, bkgrndGradient);

        errorControl = new DisplayErrorControl();

        var content = new StackLayout
        {
            Padding = new Thickness(40, 30),
            Spacing = 5,
            Children =
            {
                new Label { Text = "Geolokalizacja", Style = CustomStyles.RobotoLightTextStyle, HorizontalTextAlignment = TextAlignment.Center, VerticalOptions = LayoutOptions.Start, Margin = new Thickness(0,0,0,18) },
                new Label { Text = "Prosimy o zgodę na geolokalizację, bez tego nie ustalimy  gdzie ktoś potrzebuje pomocy.", TextColor = Color.White, FontSize = 18, HorizontalTextAlignment = TextAlignment.Center, VerticalOptions = LayoutOptions.Start},
                new CircleImage
                {
                    FillColor = Color.White,
                    BorderColor = Color.FromHex("#19ccd5"),
                    BorderThickness = 10,
                    HeightRequest = 140,
                    WidthRequest = 140,
                    Aspect = Aspect.AspectFill,
                    HorizontalOptions = LayoutOptions.Center,
                    Source = "ic_map",
```

```
            VerticalOptions = LayoutOptions.CenterAndExpand
        },
        (loader = new ActivityIndicator() {Color = Color.White}),
        (acceptButton = new Button{ Text = "WYRAŻAM ZGODĘ", Style = CustomStyles.WhiteButtonStyle, VerticalOptions = LayoutOptions.End, HeightRequest= 55, Command=vm.AccessGeolocationCommand }),
        new StackLayout
        {
            Margin = new Thickness(0,10,0,0),
            Orientation = StackOrientation.Horizontal,
            HorizontalOptions = LayoutOptions.Center,
            VerticalOptions = LayoutOptions.End,
            Spacing = 8,
            Children =
            {
                new RoundedBoxView.Forms.Plugin.Abstractions.RoundedBoxView
                {
                    BackgroundColor = Color.Transparent,
                    WidthRequest = 12,
                    HeightRequest = 12,
                    CornerRadius = 6,
                    BorderColor = Color.White,
                    BorderThickness= 1
                },
                new RoundedBoxView.Forms.Plugin.Abstractions.RoundedBoxView
                {
                    BackgroundColor = Color.Transparent,
                    WidthRequest = 12,
                    HeightRequest = 12,
                    CornerRadius = 6,
                    BorderColor = Color.White,
                    BorderThickness= 1
                },
                new RoundedBoxView.Forms.Plugin.Abstractions.RoundedBoxView
                {
```

```
                    BackgroundColor = Color.White,
                    WidthRequest = 12,
                    HeightRequest = 12,
                    CornerRadius = 6
                }
            }
        }
    };

    loader.SetBinding(ActivityIndicator.IsRunningProperty, "IsBusy");
    loader.SetBinding(ActivityIndicator.IsVisibleProperty, "IsBusy");

    acceptButton.SetBinding(Button.IsVisibleProperty, "IsBusy", converter:new
BoolConverter());

    Content = new Grid
    {
        Children =
        {
            content,
            errorControl
        }
    };
}

protected override void OnAppearing()
{
    base.OnAppearing();
    errorService.OnError += ErrorService_OnError;
}

protected override void OnDisappearing()
{
```

```
            base.OnDisappearing();
            errorService.OnError -= ErrorService_OnError;
        }

        async void ErrorService_OnError(object sender, ErrorMsg e)
        {
            await errorControl.Show(e);
        }
    }
}
```

PhotoPage:

```
using System;
using System.Diagnostics;
using ImageCircle.Forms.Plugin.Abstractions;
using PoMOST.Models;
using PoMOST.Services;
using PoMOST.Styles;
using PoMOST.ViewModels.RegistrationViewModels;
using PoMOST.Views.Controls;
using Xamarin.Forms;
using XFGloss;

namespace PoMOST.Views.Pages.Registration
{
    public class PhotoPage : ContentPage
    {
        PhotoPageViewModel vm;
        CircleImage faceImage;
        ActivityIndicator loader;
        DisplayErrorControl errorControl;
        DisplayErrorService errorService;
```

```
public PhotoPage(PostUserDto userData)
{
    NavigationPage.SetHasNavigationBar(this, false);
    errorService = ContainerAccessor.Resolve<DisplayErrorService>();

    BindingContext = vm = ContainerAccessor.Resolve<PhotoPageViewModel>();
    vm.UserData = userData;

    var bkgrndGradient = new Gradient()
    {
        Rotation = 0,
        Steps = new GradientStepCollection()
        {
            new GradientStep(Color.FromHex("#19ccd5"), 0),
            new GradientStep(Color.FromHex("#0eabfa"), 1)
        }
    };

    var text = new Label { VerticalOptions = LayoutOptions.Fill, HorizontalOptions = LayoutOptions.FillAndExpand };

    ContentPageGloss.SetBackgroundGradient(this, bkgrndGradient);

    var takePhotoButton = new Button { Text = "ZRÓB ZDJĘCIE", Style = CustomStyles.WhiteButtonStyle, VerticalOptions = LayoutOptions.EndAndExpand, Command = vm.PhotoButtonCommand };
    var galleryButton = new Button { Text = "WYBIERZ Z GALERII", Style = CustomStyles.WhiteButtonStyle, Command = vm.GalleryButtonCommand };
    var saveButton = new Button { Text = "ZAPISZ", Style = CustomStyles.WhiteButtonStyle, Command = vm.SaveButtonCommand };
    var clearButton = new Button { Text = "WYCZYŚĆ", Style = CustomStyles.WhiteButtonStyle, Command = vm.ClearButtonCommand };

    takePhotoButton.SetBinding(Button.IsVisibleProperty, "PhotoButtonsVisible");
    galleryButton.SetBinding(Button.IsVisibleProperty, "PhotoButtonsVisible");
    saveButton.SetBinding(Button.IsVisibleProperty, "UtilsButtonsVisible");
```

```
clearButton.SetBinding(Button.IsVisibleProperty, "UtilsButtonsVisible");

//inital set values
//takePhotoButton.IsVisible = true;
//galleryButton.IsVisible = true;
//saveButton.IsVisible = false;
//clearButton.IsVisible = false;

var content = new StackLayout
{
    Padding = new Thickness(40,30),
    Spacing = 5,
    Children =
    {
        new Label { Text = "Dodaj zdjęcie", Style = CustomStyles.RobotoLightTextStyle, HorizontalTextAlignment = TextAlignment.Center},
        new Label { Text = "Dodanie zdjęcia sprawi, że łatwiej będzie Ci nawiązać kontakt z innymi.", TextColor = Color.White, FontSize = 18, HorizontalTextAlignment = TextAlignment.Center},
        new StackLayout
        {
            VerticalOptions = LayoutOptions.CenterAndExpand,
            Children =
            {
                (faceImage = new CircleImage
                {
                    FillColor = Color.White,
                    BorderColor = Color.FromHex("#19ccd5"),
                    BorderThickness = 10,
                    HeightRequest = 140,
                    WidthRequest = 140,
                    Aspect = Aspect.AspectFill,
                    HorizontalOptions = LayoutOptions.Center,
                    Margin = new Thickness(0,0,0,20)
                }),
```

```
                (loader = new ActivityIndicator(){Color = Color.White}),
                takePhotoButton,
                galleryButton,
                clearButton,
                saveButton,
            }
        },
        new Button{ Text = "POMIŃ", Style = CustomStyles.WhiteButtonStyle, VerticalOptions = LayoutOptions.End, Command=vm.SkipCommand },
        new StackLayout
        {
            Margin = new Thickness(0,10,0,0),
            Orientation = StackOrientation.Horizontal,
            HorizontalOptions = LayoutOptions.Center,
            VerticalOptions = LayoutOptions.End,
            Spacing = 8,
            Children =
            {
                new RoundedBoxView.Forms.Plugin.Abstractions.RoundedBoxView
                {
                    BackgroundColor = Color.Transparent,
                    WidthRequest = 12,
                    HeightRequest = 12,
                    CornerRadius = 6,
                    BorderColor = Color.White,
                    BorderThickness= 1
                },
                new RoundedBoxView.Forms.Plugin.Abstractions.RoundedBoxView
                {
                    BackgroundColor = Color.White,
                    WidthRequest = 12,
                    HeightRequest = 12,
                    CornerRadius = 6
                },
```

```
                new RoundedBoxView.Forms.Plugin.Abstractions.RoundedBoxView
                {
                    BackgroundColor = Color.Transparent,
                    WidthRequest = 12,
                    HeightRequest = 12,
                    CornerRadius = 6,
                    BorderColor = Color.White,
                    BorderThickness= 1
                }
            }
        }
    }
};
errorControl = new DisplayErrorControl();
Content = new Grid
{
    Children =
    {
        content,
        errorControl
    }
};

loader.SetBinding(ActivityIndicator.IsRunningProperty, "IsBusy");
loader.SetBinding(ActivityIndicator.IsVisibleProperty, "IsBusy");
faceImage.SetBinding(CircleImage.SourceProperty, "PhotoSource");
}

protected override void OnAppearing()
{
    base.OnAppearing();
    errorService.OnError += ErrorService_OnError;
}
```

```
        protected override void OnDisappearing()
        {
            base.OnDisappearing();
            errorService.OnError -= ErrorService_OnError;
        }

        async void ErrorService_OnError(object sender, ErrorMsg e)
        {
            await errorControl.Show(e);
        }
    }
}
```

Popups:

SexPopup:

```
using System;
using System.Threading.Tasks;
using PoMOST.Services.Navigation;
using PoMOST.Styles;
using PoMOST.ViewModels.RegistrationViewModels.RegistrationPopupsViewModels;
using PoMOST.Views.Popups;
using Xamarin.Forms;

namespace PoMOST.Views.Pages.Registration.Popups
{
    public class SexPopup : BasicNoBoxPopup
    {
        SexPopupViewModel vm;

        Button manButton;
        Button womanButton;
```

```
        public SexPopup()
        {
            vm = ContainerAccessor.Resolve<SexPopupViewModel>();

            var navigationService = ContainerAccessor.Resolve<INavigationService>();

            PopupContent.Content = new StackLayout
            {
                Spacing = 5,
                Children =
                {
                    (manButton = new Button { Text = "Kobieta", Style = CustomStyles.WhiteButtonStyle
, VerticalOptions = LayoutOptions.EndAndExpand, HeightRequest = 60  }),
                    (womanButton = new Button { Text = "Mężczyzna", Style =
CustomStyles.WhiteButtonStyle, HeightRequest = 60  })
                }
            };

            manButton.Clicked += (sender, e) =>
            {
                Value = "Kobieta";
                navigationService.ClosePopup();
            };

            womanButton.Clicked += (sender, e) =>
            {
                Value = "Mężczyzna";
                navigationService.ClosePopup();
            };
        }


    }
}
```

UserPopup:

```
using System;
using PoMOST.Services.Navigation;
using PoMOST.Styles;
using PoMOST.ViewModels.RegistrationViewModels.RegistrationPopupsViewModels;
using PoMOST.Views.Popups;
using Xamarin.Forms;

namespace PoMOST.Views.Pages.Registration.Popups
{
   public class UserPopup : BasicNoBoxPopup
   {
      UserPopupViewModel vm;
      Button seniorButton;
      Button helperButton;
      Button boothButton;

      public UserPopup()
      {
         vm = ContainerAccessor.Resolve<UserPopupViewModel>();

         var navigationService = ContainerAccessor.Resolve<INavigationService>();

         PopupContent.Content = new StackLayout
         {
            Spacing = 5,
            Children =
            {
               (seniorButton = new Button { Text = "Potrzebuję pomocy", Style = CustomStyles.WhiteButtonStyle , VerticalOptions = LayoutOptions.EndAndExpand, HeightRequest = 60 }),
               (helperButton = new Button { Text = "Chcę pomagać", Style = CustomStyles.WhiteButtonStyle, HeightRequest = 60 }),
```

```
            (boothButton = new Button { Text = "Jedno i drugie", Style =
CustomStyles.WhiteButtonStyle, HeightRequest = 60 })
            }
        };

        seniorButton.Clicked += (sender, e) =>
        {
            Value = "Potrzebuję pomocy";
            navigationService.ClosePopup();
        };

        helperButton.Clicked += (sender, e) =>
        {
            Value = "Chcę pomagać";
            navigationService.ClosePopup();
        };

        boothButton.Clicked += (sender, e) =>
        {
            Value = "Jedno i drugie";
            navigationService.ClosePopup();
        };
    }
  }
}
```

RegistrationPage:

```
using System;
using PoMOST.Models;
using PoMOST.Services;
using PoMOST.Styles;
using PoMOST.ViewModels.RegistrationViewModels;
using PoMOST.Views.Controls;
```

```
using PoMOST.Views.Pages.Registration.Popups;
using Xamarin.Forms;
using XFGloss;

namespace PoMOST.Views.Pages.Registration
{
    public class RegistrationPage : ContentPage
    {
        RegistrationPageViewModel vm;
        Button nextButton;
        RegistrationEntry nameEntry;
        RegistrationEntry phoneEntry;
        RegistrationUserButton<UserPopup> rolePicker;
        RegistrationUserButton<SexPopup> genderPicker;
        DisplayErrorControl errorControl;
        DisplayErrorService errorService;

        public RegistrationPage()
        {
            NavigationPage.SetHasNavigationBar(this, false);

            errorService = ContainerAccessor.Resolve<DisplayErrorService>();

            vm = ContainerAccessor.Resolve<RegistrationPageViewModel>();

            var bkgrndGradient = new Gradient()
            {
                Rotation = 0,
                Steps = new GradientStepCollection()
                {
                    new GradientStep(Color.FromHex("#19ccd5"), 0),
                    new GradientStep(Color.FromHex("#0eabfa"), 1)
                }
            };
```

```
var text = new Label { VerticalOptions = LayoutOptions.Fill, HorizontalOptions = LayoutOptions.FillAndExpand };


ContentPageGloss.SetBackgroundGradient(this, bkgrndGradient);



var content = new StackLayout
{
   Padding = new Thickness(40,30),
   Children =
   {
      new Label { Text = "Zarejestruj się", Style = CustomStyles.RobotoLightTextStyle, HorizontalTextAlignment = TextAlignment.Center, Margin = new Thickness(0,16,0,0)},
      new Label { Text = "To proste. Prosimy o podanie kilku podstawowych informacji.", TextColor = Color.White, FontSize = 18, HorizontalTextAlignment = TextAlignment.Center, VerticalOptions = LayoutOptions.StartAndExpand, Margin = new Thickness(0,18,0,0) },
      new Grid
      {
         //Margin = new Thickness(0,100,0,0), + scrollView
         Padding = 0,
         Children =
         {
            new RoundedBoxView.Forms.Plugin.Abstractions.RoundedBoxView(){ BackgroundColor = Color.White, CornerRadius = 5, BorderColor = Color.LightBlue, BorderThickness = 0 },
            text,
            new StackLayout
            {
               Spacing = 0,
               Children =
               {
                  (nameEntry = new RegistrationEntry("Imię", Keyboard.Text)),
                  new BoxView { BackgroundColor = Color.FromHex("#330da9fd") , HeightRequest = 1 },
                  (phoneEntry = new RegistrationEntry("Numer telefonu", Keyboard.Telephone)),
```

```
                    new BoxView { BackgroundColor = Color.FromHex("#330da9fd") , HeightRequest = 1 },
                    (rolePicker = new RegistrationUserButton<UserPopup>("Kim jesteś?",0 )),
                    new BoxView { BackgroundColor = Color.FromHex("#330da9fd") , HeightRequest = 1 },
                    (genderPicker = new RegistrationUserButton<SexPopup>("Płeć",1 ))
                }
            }
        }
    },
    (nextButton = new Button{ Text = "DALEJ", Style = CustomStyles.WhiteButtonStyle, Margin = new Thickness(0,30,0,0) }),
    new StackLayout
    {
        Margin = new Thickness(0,10,0,0),
        Orientation = StackOrientation.Horizontal,
        HorizontalOptions = LayoutOptions.Center,
        Spacing = 8,
        Children =
        {
            new RoundedBoxView.Forms.Plugin.Abstractions.RoundedBoxView
            {
                BackgroundColor = Color.White,
                WidthRequest = 12,
                HeightRequest = 12,
                CornerRadius = 6
            },
            new RoundedBoxView.Forms.Plugin.Abstractions.RoundedBoxView
            {
                BackgroundColor = Color.Transparent,
                WidthRequest = 12,
                HeightRequest = 12,
                CornerRadius = 6,
                BorderColor = Color.White,
                BorderThickness= 1
```

```
            },
            new RoundedBoxView.Forms.Plugin.Abstractions.RoundedBoxView
            {
                BackgroundColor = Color.Transparent,
                WidthRequest = 12,
                HeightRequest = 12,
                CornerRadius = 6,
                BorderColor = Color.White,
                BorderThickness= 1
            }
        }
    }
  }
};

errorControl = new DisplayErrorControl();

Content = new Grid
{
  Children =
  {
    content,
    errorControl
  }
};

nextButton.Clicked += (sender, e) =>
{
  var isValidate = true;
  if(!vm.ValidateName(nameEntry.Text))
  {
    nameEntry.ShowValidateLabel();
    isValidate = false;
  }
```

```
            if(!vm.ValidatePhone(phoneEntry.Text))
            {
                phoneEntry.ShowValidateLabel();
                isValidate = false;
            }
            if(!vm.ValidateRole(rolePicker.Value))
            {
                rolePicker.ShowValidateLabel();
                isValidate = false;
            }
            if(!vm.ValidateGender(genderPicker.Value))
            {
                genderPicker.ShowValidateLabel();
                isValidate = false;
            }

            if (!isValidate)
            {
                errorControl.Show(new ErrorMsg {Text= "Uzupełnij wszystkie dane.", Success = false
});
                return;
            }

            var helper = false;
            var needy = false;
            switch (rolePicker.Value)
            {
                case "Chcę pomagać":
                    helper = true;
                    needy = false;
                    break;
                case "Potrzebuję pomocy":
                    helper = false;
                    needy = true;
```

```
                break;
            case "Jedno i drugie":
                helper = true;
                needy = true;
                break;
        }

        var gender = "FEMALE";
        switch (genderPicker.Value)
        {
            case "Kobieta":
                gender = "FEMALE";
                break;
            case "Mężczyzna":
                gender = "MALE";
                break;
        }

        var userData = new PostUserDto()
        {
            Name = nameEntry.Text,
            Phone = phoneEntry.Text,
            Needy = needy,
            Helper = helper,
            Gender = gender
        };

        vm.UserData = userData;
        vm.NextStepCommand.Execute(null);
    };
}

protected override void OnAppearing()
{
```

```
            base.OnAppearing();
            errorService.OnError += ErrorService_OnError;
        }

        protected override void OnDisappearing()
        {
            base.OnDisappearing();
            errorService.OnError -= ErrorService_OnError;
        }

        async void ErrorService_OnError(object sender, ErrorMsg e)
        {
            await errorControl.Show(e);
        }
    }
}
```

SuccessPage:

```
using System;
using ImageCircle.Forms.Plugin.Abstractions;
using PoMOST.Styles;
using PoMOST.ViewModels.RegistrationViewModels;
using Xamarin.Forms;
using XFGloss;

namespace PoMOST.Views.Pages.Registration
{
    public class SuccessPage : ContentPage
    {
        SuccessPageViewModel vm;
        ActivityIndicator loader;

        public SuccessPage()
```

```
{
    NavigationPage.SetHasNavigationBar(this, false);
    BindingContext = vm = ContainerAccessor.Resolve<SuccessPageViewModel>();

    var bkgrndGradient = new Gradient()
    {
        Rotation = 0,
        Steps = new GradientStepCollection()
        {
            new GradientStep(Color.FromHex("#19ccd5"), 0),
            new GradientStep(Color.FromHex("#0eabfa"), 1)
        }
    };

    var text = new Label { VerticalOptions = LayoutOptions.Fill, HorizontalOptions = LayoutOptions.FillAndExpand };

    ContentPageGloss.SetBackgroundGradient(this, bkgrndGradient);

    Content = new StackLayout
    {
        Padding = new Thickness(40,30),
        Spacing = 5,
        Children =
        {
            new Label { Text = "Udało się!", Style = CustomStyles.RobotoLightTextStyle, HorizontalTextAlignment = TextAlignment.Center, VerticalOptions = LayoutOptions.Start, Margin = new Thickness(0,0,0,18)},
            new Label { Text = "Możesz już używać PoMOSTu :)", TextColor = Color.White, FontSize = 18, HorizontalTextAlignment = TextAlignment.Center, VerticalOptions = LayoutOptions.Start },
            new CircleImage
            {
                Margin = new Thickness(0,22,0,0),
                FillColor = Color.White,
                BorderColor = Color.FromHex("#19ccd5"),
```

```
                BorderThickness = 10,
                HeightRequest = 140,
                WidthRequest = 140,
                Aspect = Aspect.AspectFill,
                HorizontalOptions = LayoutOptions.Center,
                Source = "ic_ok",
                VerticalOptions = LayoutOptions.CenterAndExpand
            },
            (loader = new ActivityIndicator(){Color = Color.White, VerticalOptions = LayoutOptions.End}),
            new Button{ Text = "ZACZYNAMY!", Style = CustomStyles.WhiteButtonStyle, VerticalOptions = LayoutOptions.End, HeightRequest = 55, Command = vm.CompleteRegistration }
        }
    };

    loader.SetBinding(ActivityIndicator.IsRunningProperty, "IsBusy");
    loader.SetBinding(ActivityIndicator.IsVisibleProperty, "IsBusy");
  }
 }
}
```

SettingsPage:

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using FFImageLoading.Svg.Forms;
using ImageCircle.Forms.Plugin.Abstractions;
using PoMOST.Helpers;
using PoMOST.Styles;
using PoMOST.ViewModels;
using PoMOST.Views.Controls;
using Xamarin.Forms;
using XFGloss;
```

```
namespace PoMOST.Views.Pages
{
   public class SettingsPage : BasicPage
   {
      SettingsPageViewModel vm;

      SettingsPageButton nameButton;
      SettingsPageButton phoneButton;

      List<int> Ranges = new List<int>() { 100, 300, 500, 1000, 1500, 2000, 3000, 5000, 10000};

      GradientBlueButton saveAvatarButton;
      GradientBlueButton saveDescriptionButton;
      GradientBlueButton saveOptionsButton;

      public SettingsPage()
      {
         Title = "USTAWIENIA";

         NavigationPage.SetHasNavigationBar(this, false);

         BindingContext = vm = ContainerAccessor.Resolve<SettingsPageViewModel>();

         var relativeLayout = new RelativeLayout();

         var borderEntry = new BorderEntryControl("Mój opis") { HeightRequest = 140, Margin =
new Thickness(0, 20, 0, 0) };
         borderEntry.entry.TextChanged += (sender, e) =>
         {
            if (e.NewTextValue == borderEntry.placeholder)
            {
               vm.TextCounter = 0;
               return;
```

```
        }
        vm.TextCounter = e.NewTextValue.Length;
        borderEntry.Text = e.NewTextValue;

    };

    borderEntry.SetBinding(BorderEntryControl.TextProperty, "UserDescription");

    var textCountLabel = new Label { Text = $"Pozostało znaków", Style = CustomStyles.TextPopupStyle, HorizontalTextAlignment = TextAlignment.End , Margin = new Thickness(0, 0, 0, 0), FontSize = 14 };
    textCountLabel.SetBinding(Label.TextProperty, "TextCounter", stringFormat: "Pozostało {0} znaków");

    var text = new Label { VerticalOptions = LayoutOptions.Fill, HorizontalOptions = LayoutOptions.FillAndExpand };

    var rangeSlider = new Slider { Minimum = 0, Maximum = 8, Margin = new Thickness(0, 25, 0, 0) };
    SliderGloss.SetMinTrackTintColor(rangeSlider,CustomStyles.DarkBlue);
    SliderGloss.SetMaxTrackTintColor(rangeSlider, CustomStyles.DarkBlue);
    SliderGloss.SetThumbTintColor(rangeSlider, CustomStyles.LightBlue);

    rangeSlider.SetBinding(Slider.ValueProperty, "SliderDistance");

    rangeSlider.ValueChanged += (object sender, ValueChangedEventArgs e) =>
    {
        var newStep = Math.Round(e.NewValue / 1);
        rangeSlider.Value = newStep * 1;
        var newValue = Ranges[(int)e.NewValue];
        vm.UserDistance = newValue;
    };

    var rangeLabel = new Label
    {
        Style = CustomStyles.ProfileDarkTextRegularStyle
```

```
        };
        rangeLabel.SetBinding(Label.TextProperty, "UserDistance", stringFormat: "{0}m");

        var notificationCell = new Switch{ Margin = new Thickness(0,0,10,0) };
        notificationCell.SetBinding(Switch.IsToggledProperty, "UserNotification");

        var circleImage = new CircleImage
        {
            FillColor = Color.White,
            BorderColor = Color.FromHex("#19ccd5"),
            BorderThickness = 10,
            HeightRequest = 140,
            WidthRequest = 140,
            Aspect = Aspect.AspectFill,
            HorizontalOptions = LayoutOptions.Center,
        };
        circleImage.SetBinding(CircleImage.SourceProperty, "PhotoSource");

        var addPhotoIcon = new SvgCachedImage { Source = SvgImageSource.FromFile("aparat"), VerticalOptions = LayoutOptions.CenterAndExpand, HeightRequest = 40, WidthRequest = 40, Margin = new Thickness(10, 13, 10, 15) };
        //addPhotoIcon.GestureRecognizers.Add(vm.PhotoTap);

        relativeLayout.Children.Add(
            circleImage,
            Constraint.RelativeToParent((parent) =>
            {
            return (parent.Width / 2) - 70;
            }),
            Constraint.RelativeToParent((parent) =>
            {
                return 0;
            }),
            Constraint.Constant(140),
            Constraint.Constant(140)
```

```
);

/*relativeLayout.Children.Add(
    addPhotoIcon,
    Constraint.RelativeToView(circleImage, (parent, sibling) => {
    return (parent.Width / 2) ;
    }),
    Constraint.RelativeToView(circleImage, (parent, sibling) => {
        return 2 * sibling.Height / 4;
    }),
    Constraint.Constant(100),
    Constraint.Constant(100)
    );*/

PageContent.Content = new ScrollView
{
    Content = new StackLayout
    {
        Spacing = 0,
        Children = {
            relativeLayout,
            //(saveAvatarButton =new GradientBlueButton(){ Text = "DODAJ ZDJĘCIE", Margin = new Thickness(0,-10,0,10), Command=vm.SaveAvatarCommand }),
            new Label { Text = "Moje dane", Style = CustomStyles.ProfileDarkTextMediumStyle, Margin = new Thickness(10,10,10,10) },
            //W rounded box

            new Grid
            {
                Children =
                {
                    new RoundedBoxView.Forms.Plugin.Abstractions.RoundedBoxView(){ BackgroundColor = Color.White, CornerRadius = 5, BorderColor = Color.LightBlue, BorderThickness = 1 },
                    text,
```

```
                new StackLayout
                {
                    Spacing = 0,
                    Padding = 1,
                    Children =
                    {
                        (nameButton = new SettingsPageButton("Imię", "", "popraw", "profil", 0, vm.OpenSettingsCommand)),
                        new BoxView { BackgroundColor = Color.FromHex("#330da9fd") , HeightRequest = 2 },
                        (phoneButton = new SettingsPageButton("Telefon", "","zmień", "telefon", 1, vm.OpenSettingsCommand))
                    }
                }
            },
            borderEntry,
            textCountLabel,
            (saveDescriptionButton = new GradientBlueButton(){ Text = "ZAPISZ OPIS", Margin = new Thickness(0,10,0,0), Command=vm.SaveDescriptionCommand }),
            new Label { Text = "Ustawienia próśb", Style = CustomStyles.ProfileDarkTextMediumStyle, Margin = new Thickness(0,40,0,0) },
            new BoxView { HeightRequest = 2, BackgroundColor = Color.FromHex("#330da9fd") , Margin = new Thickness(0,10,0,0) },
            new StackLayout
            {
                Orientation = StackOrientation.Horizontal,
                Margin = new Thickness(0,14,0,0),
                Children =
                {
                    new Label { Text = "Maksymalna odległość", HorizontalOptions = LayoutOptions.StartAndExpand, Style = CustomStyles.ProfileDarkTextRegularStyle },
                    rangeLabel
                }
            },
            rangeSlider,
```

```
new BoxView { HeightRequest = 2, BackgroundColor = Color.FromHex("#330da9fd"), Margin = new Thickness(0,15,0,0) },
new StackLayout
{
  Orientation = StackOrientation.Horizontal,
  Children =
  {
    new SvgCachedImage { Source = SvgImageSource.FromFile("powiadomienia"), VerticalOptions = LayoutOptions.CenterAndExpand, HeightRequest = 40, WidthRequest = 40 , Margin = new Thickness(10,13,10,15) },
    new Label { Text = "Powiadomienia", Style = CustomStyles.ProfileDarkTextRegularStyle, VerticalOptions = LayoutOptions.Center , HorizontalOptions = LayoutOptions.StartAndExpand },
    notificationCell
  }
},
new BoxView { HeightRequest = 2, BackgroundColor = Color.FromHex("#330da9fd") },
(saveOptionsButton = new GradientBlueButton(){ Text = "ZAPISZ USTAWIENIA", Margin = new Thickness(0,10,0,0), Command=vm.SaveNotificationSettingsCommand }),
new Label { Text = "Regulamin i autorzy", Style = CustomStyles.ProfileDarkTextMediumStyle, Margin = new Thickness(0,20,0,0) },
new Grid
{
  Margin = new Thickness(0,10,0,0),
  Children =
  {
    new RoundedBoxView.Forms.Plugin.Abstractions.RoundedBoxView(){ BackgroundColor = Color.White, CornerRadius = 5, BorderColor = Color.LightBlue, BorderThickness = 1 },
    text,
    new StackLayout
    {
      Spacing = 0,
      Padding = 1,
      Children =
```

```
                    {
                        new SettingsAboutButton("Regulamin", "zobacz", "regulamin", 2, vm.OpenSettingsCommand),
                        new BoxView { BackgroundColor = Color.FromHex("#330da9fd") , HeightRequest = 2 },
                        new SettingsAboutButton("O autorach", "zobacz", "autorzy", 3, vm.OpenSettingsCommand),
                    }
                }
            }
        },

    }
  }
};

if(App.ROLE=="HELPER")
{
  borderEntry.IsVisible = false;
  textCountLabel.IsVisible = false;
  saveDescriptionButton.IsVisible = false;
}

//saveAvatarButton.SetBinding(Button.IsEnabledProperty, "IsBusy", converter:new BoolConverter());
saveDescriptionButton.SetBinding(Button.IsEnabledProperty, "IsBusy", converter: new BoolConverter());
saveOptionsButton.SetBinding(Button.IsEnabledProperty, "IsBusy", converter: new BoolConverter());

nameButton.SetBinding(SettingsPageButton.TextProperty, "UserName");
phoneButton.SetBinding(SettingsPageButton.TextProperty, "UserPhone");
}

protected override void OnAppearing()
{
```

```
            base.OnAppearing();
            vm.OnAppear();
        }

        protected override void OnDisappearing()
        {
            base.OnDisappearing();
            vm.OnDisappear();
        }
    }
}
```

TermsPage:

```
using System;
using FFImageLoading.Svg.Forms;
using PoMOST.Styles;
using PoMOST.ViewModels;
using Xamarin.Forms;

namespace PoMOST.Views.Pages
{
    public class TermsPage : BasicModalPage
    {
        public TermsPage()
        {
            Title = "REGULAMIN";
            vm = ContainerAccessor.Resolve<TermsPageViewModel>();

            PageContent.Content = new ScrollView
            {
                Content = new StackLayout
                {
```

```
                Padding = new Thickness(10,0,10,10),
                Children =
                {
                    new SvgCachedImage { Source = SvgImageSource.FromFile("regulamin"), VerticalOptions = LayoutOptions.Center, HeightRequest = 80, WidthRequest = 80 },
                    new Label { Style = CustomStyles.ProfileDarkTextMediumStyle, HorizontalTextAlignment=TextAlignment.Center, FontSize = 16, Text = "Regulamin korzystania z aplikacji mobilnej poMOST – wspierającej aktywizację i niezależność osób starszych", Margin = new Thickness(0,15,0,0) },
                    new Label { Style = CustomStyles.ProfileDarkTextRegularStyle, FontSize = 16, Text = "", Margin = new Thickness(0,15,0,0) }
                }
            }
        };
    }
  }
}
```

VoiceCommandsPage:

```
using System;
using FFImageLoading.Svg.Forms;
using PoMOST.Styles;
using PoMOST.ViewModels;
using PoMOST.Views.Controls;
using Xamarin.Forms;

namespace PoMOST.Views.Pages
{
  public class VoiceCommandsPage : BasicPage
  {
    VoiceCommandsPageViewModel vm;

    public VoiceCommandsPage()
    {
```

```
Title = "TRYB GŁOSOWY";

NavigationPage.SetHasNavigationBar(this, false);

vm = ContainerAccessor.Resolve<VoiceCommandsPageViewModel>();

var grid = new Grid
{
   Margin = new Thickness(33,14,33,0),
   ColumnDefinitions =
   {
      new ColumnDefinition { Width = new GridLength(0.5,GridUnitType.Star) },
      new ColumnDefinition { Width = new GridLength(1,GridUnitType.Auto) }
   }
};

var leftLabel = new Label
{
   Text ="•  Zakupy\n•  Zdrowie\n•  Transport\n•  Poczta",
   Style = CustomStyles.ProfileDarkTextRegularStyle,
   FontSize = 16
};

var rightLabel = new Label
{
   Text = "•  Pomocy\n•  Komputer\n•  Inne",
   Style = CustomStyles.ProfileDarkTextRegularStyle,
   FontSize = 16
};

grid.Children.Add(leftLabel,0,0);
grid.Children.Add(rightLabel, 1, 0);

PageContent.Content = new StackLayout
```

```
        {
            Children = {
                new SvgCachedImage { Source = SvgImageSource.FromFile("mic_ic"), VerticalOptions = LayoutOptions.Center, HeightRequest = 80, WidthRequest = 80, Margin = new Thickness(10,0,20,0) },
                new Label { Text = "By rozpocząć, proszę wcisnąć przycisk \"rozpocznij\" i powiedzieć jedno z następujących słów:", Style = CustomStyles.ProfileDarkTextRegularStyle, FontSize = 16, Margin = new Thickness(23,15,0,0) },
                grid,
                new GradientBlueButton { Text = "ROZPOCZNIJ", Margin = new Thickness(0,20,0,18), Command = vm.startCommand, VerticalOptions = LayoutOptions.EndAndExpand }
            }
        };
    }
  }
}
```

Popups

AlarmRequestPopup:

```
using System;
using FFImageLoading.Svg.Forms;
using PoMOST.Models;
using PoMOST.Styles;
using PoMOST.ViewModels.PopupsViewModels;
using PoMOST.Views.Controls;
using Xamarin.Forms;

namespace PoMOST.Views.Popups
{
  public class AlarmRequestPopup : BasicPopup
  {
    HelpRequestPopupViewModel vm;
    GradientBlueButton sendButton;
```

```
public AlarmRequestPopup()
{
    BindingContext = vm = ContainerAccessor.Resolve<HelpRequestPopupViewModel>();

    var borderEntry = new BorderEntryControl("Dodatkowe informacje") { HeightRequest = 160 };
    borderEntry.entry.TextChanged += (sender, e) =>
    {
        if (e.NewTextValue == borderEntry.placeholder)
        {
            vm.TextCounter = 0;
            return;
        }
        vm.TextCounter = e.NewTextValue.Length;
    };

    var textCountLabel = new Label { Text = $"Pozostało znaków", Style = CustomStyles.TextPopupStyle, HorizontalTextAlignment = TextAlignment.End, FontSize = 14, Margin = new Thickness(0, -12, 0, 0) };
    textCountLabel.SetBinding(Label.TextProperty, "TextCounter", stringFormat: "Pozostało {0} znaków");

    PopupContent.Content = new StackLayout
    {
        Spacing = 10,
        Children =
        {

            new SvgCachedImage{ WidthRequest=80, HeightRequest=80, Source = SvgImageSource.FromFile("alarm"), HorizontalOptions = LayoutOptions.Center, VerticalOptions = LayoutOptions.Center },
            new Label { Text = "ALARM" , Style = CustomStyles.TitlePopupStyle, HorizontalTextAlignment = TextAlignment.Center },
            new Label { Text = "Czy wysłać wiadomość, że potrzebujesz pilnej pomocy lub jesteś w niebezpieczeństwie?" , Style = CustomStyles.TextPopupStyle },
            borderEntry,
            textCountLabel,
```

```
            new StackLayout
            {
                Spacing = 20,
                Orientation = StackOrientation.Horizontal,
                Children =
                {
                    new Button{ Text = "ANULUJ", Command=vm.CloseCommand, Style = CustomStyles.WhiteButtonStyle ,HorizontalOptions = LayoutOptions.FillAndExpand },
                    (sendButton = new GradientBlueButton { Text = "WYŚLIJ", HorizontalOptions = LayoutOptions.FillAndExpand})
                }
            }
        };

        sendButton.Clicked += async (sender, e) =>
        {
            var msg = "jest w niebezpieczeństwie i pilnie potrzebuje pomocy! Skontaktuj się jak najszybciej.";
            if (!string.IsNullOrEmpty(borderEntry.CurrentText))
            {
                msg += "\n\n";
                msg += borderEntry.CurrentText;
            }
            await vm.SendHelpRequest(msg, HelpRequest.Categories.ALERT.ToString());
        };
    }
  }
}
```

BasicNoBoxPopup:

```
using System;
using System.Diagnostics;
using System.Threading.Tasks;
```

```
using Rg.Plugins.Popup.Pages;
using Xamarin.Forms;

namespace PoMOST.Views.Popups
{
   public class BasicNoBoxPopup : PopupPage
   {
      public ContentView PopupContent;
      public Task<string> PopupClosedTask => tcs.Task;
      public string Value { get; set; }
      private TaskCompletionSource<string> tcs { get; set; }

      public BasicNoBoxPopup()
      {
         this.BackgroundColor = Color.FromHex("#b20eabfa");
         this.HasSystemPadding = true;
         this.CloseWhenBackgroundIsClicked = true;

         tcs = new TaskCompletionSource<string>();


         Content = new Grid
         {
            VerticalOptions = LayoutOptions.End,
            HorizontalOptions = LayoutOptions.FillAndExpand,
            Children =
            {
               (PopupContent = new ContentView() { Padding = 10 })
            }
         };
      }

      protected override void OnDisappearing()
      {
```

```
            base.OnDisappearing();
            Debug.WriteLine("eee " + Value);
            tcs.SetResult(Value);

        }
    }
}
```

BasicPopup:

```
using System;
using System.Diagnostics;
using Rg.Plugins.Popup.Pages;
using Xamarin.Forms;

namespace PoMOST.Views.Popups
{
    public class BasicPopup : PopupPage
    {
        public ContentView PopupContent;

        public BasicPopup()
        {
            this.BackgroundColor = Color.FromHex("#b20eabfa");
            this.HasSystemPadding = true;
            this.CloseWhenBackgroundIsClicked = true;

            Content = new ScrollView
            {
                HeightRequest =400,
                Content=new Grid
                {
                    Margin = new Thickness(20, 40),
```

```
                VerticalOptions = LayoutOptions.Center,
                HorizontalOptions = LayoutOptions.FillAndExpand,
                Children = {
                    new
RoundedBoxView.Forms.Plugin.Abstractions.RoundedBoxView(){BackgroundColor=Color.White
, CornerRadius=5, HorizontalOptions = LayoutOptions.FillAndExpand},
                    (PopupContent = new ContentView() { Padding = 20 })

                }
            }
        };
    }
  }
}
```

HealthRequestPopup:

```
using System;
using System.Diagnostics;
using FFImageLoading.Svg.Forms;
using PoMOST.Models;
using PoMOST.Styles;
using PoMOST.ViewModels.PopupsViewModels;
using PoMOST.Views.Controls;
using PoMOST.Views.Controls.Radio;
using Xamarin.Forms;

namespace PoMOST.Views.Popups
{
  public class HealthRequestPopup : BasicPopup
  {
    HelpRequestPopupViewModel vm;
    private RadioGroup radioGroup;
    GradientBlueButton sendButton;
```

```
public HealthRequestPopup()
{
    BindingContext = vm = ContainerAccessor.Resolve<HelpRequestPopupViewModel>();

    var radio1 = new RadioButton("Realizacja recepty", 0);
    var radio2 = new RadioButton("Umówienie wizyty", 1);

    var radioGroup = new RadioGroup();
    radioGroup.AddRadioButton(radio1);
    radioGroup.AddRadioButton(radio2);
    radioGroup.Select(radio1);

    var borderEntry = new BorderEntryControl("Dodatkowe informacje") { HeightRequest = 160 };
    borderEntry.entry.TextChanged += (sender, e) =>
    {
        if (e.NewTextValue == borderEntry.placeholder)
        {
            vm.TextCounter = 0;
            return;
        }
        vm.TextCounter = e.NewTextValue.Length;
    };

    var textCountLabel = new Label { Text = $"Pozostało znaków", Style = CustomStyles.TextPopupStyle, HorizontalTextAlignment = TextAlignment.End, FontSize = 14, Margin = new Thickness(0, -12, 0, 0) };
    textCountLabel.SetBinding(Label.TextProperty, "TextCounter", stringFormat: "Pozostało {0} znaków");

    PopupContent.Content = new StackLayout
    {
        Spacing = 10,
        Children =
        {
```

```
            new SvgCachedImage{ WidthRequest=80, HeightRequest=80, Source = SvgImageSource.FromFile("zdrowie"), HorizontalOptions = LayoutOptions.Center, VerticalOptions = LayoutOptions.Center },
            new Label { Text = "ZDROWIE" , Style = CustomStyles.TitlePopupStyle, HorizontalTextAlignment = TextAlignment.Center },
            radio1,
            radio2,
            borderEntry,
            textCountLabel,
            new StackLayout
            {
               Spacing = 20,
               Orientation = StackOrientation.Horizontal,
               Children =
               {
                  new Button{ Text = "ANULUJ", Command=vm.CloseCommand, Style = CustomStyles.WhiteButtonStyle, HorizontalOptions = LayoutOptions.FillAndExpand },
                  (sendButton = new GradientBlueButton { Text = "WYŚLIJ", HorizontalOptions = LayoutOptions.FillAndExpand})
               }
            }
         }
      };

      sendButton.Clicked += async (sender, e) =>
      {
         var msg = "";
         if (radioGroup.CurrentSelected.Id == 0)
         {
            msg = "potrzebuje wsparcia przy zakupie leków.";
         }
         else
         {
            msg = "potrzebuje pomocy przy zarejestrowaniu się do lekarza.";
         }
         Debug.WriteLine("Text " + borderEntry.CurrentText);
```

```
            if (!string.IsNullOrEmpty(borderEntry.CurrentText))
            {
                msg += "\n\n";
                msg += borderEntry.CurrentText;
            }
            Debug.WriteLine(msg);
            await vm.SendHelpRequest(msg, HelpRequest.Categories.HEALTH.ToString());
        };
    }
  }
}
```

InternetLearnRequestPopup:

```
using System;
using FFImageLoading.Svg.Forms;
using PoMOST.Models;
using PoMOST.Styles;
using PoMOST.ViewModels.PopupsViewModels;
using PoMOST.Views.Controls;
using Xamarin.Forms;

namespace PoMOST.Views.Popups
{
  public class InternetLearnRequestPopup : BasicPopup
  {
    HelpRequestPopupViewModel vm;
    GradientBlueButton sendButton;

    public InternetLearnRequestPopup()
    {
      BindingContext = vm = ContainerAccessor.Resolve<HelpRequestPopupViewModel>();
```

```
var borderEntry = new BorderEntryControl("Dodatkowe informacje") { HeightRequest = 160 };
borderEntry.entry.TextChanged += (sender, e) =>
{
   if (e.NewTextValue == borderEntry.placeholder)
   {
      vm.TextCounter = 0;
      return;
   }
   vm.TextCounter = e.NewTextValue.Length;
};

var textCountLabel = new Label { Text = $"Pozostało znaków", Style = CustomStyles.TextPopupStyle, HorizontalTextAlignment = TextAlignment.End, FontSize = 14, Margin = new Thickness(0, -12, 0, 0) };
textCountLabel.SetBinding(Label.TextProperty, "TextCounter", stringFormat: "Pozostało {0} znaków");

PopupContent.Content = new StackLayout
{
   Spacing = 10,
   Children =
   {
      new SvgCachedImage{ WidthRequest=80, HeightRequest=80, Source = SvgImageSource.FromFile("nauka_internetu"), HorizontalOptions = LayoutOptions.Center, VerticalOptions = LayoutOptions.Center },
      new Label { Text = "INTERNET I KOMPUTER" , Style = CustomStyles.TitlePopupStyle, HorizontalTextAlignment = TextAlignment.Center},
      borderEntry,
      textCountLabel,
      new StackLayout
      {
         Spacing = 20,
         Orientation = StackOrientation.Horizontal,
         Children =
         {
```

```
                new Button{ Text = "ANULUJ", Command=vm.CloseCommand, Style = CustomStyles.WhiteButtonStyle ,HorizontalOptions = LayoutOptions.FillAndExpand },
                (sendButton = new GradientBlueButton { Text = "WYŚLIJ", HorizontalOptions = LayoutOptions.FillAndExpand})
            }
        }
    };

    sendButton.Clicked += async (sender, e) =>
    {
        var msg = "potrzebuje pomocy z obsługą komputera lub telefonu komórkowego lub Internetu.";
        if (!string.IsNullOrEmpty(borderEntry.CurrentText))
        {
            msg += "\n\n";
            msg += borderEntry.CurrentText;
        }
        await vm.SendHelpRequest(msg, HelpRequest.Categories.IT.ToString());
    };
  }
 }
}
```

OtherRequestPopup:

```
using System;
using FFImageLoading.Svg.Forms;
using PoMOST.Models;
using PoMOST.Styles;
using PoMOST.ViewModels.PopupsViewModels;
using PoMOST.Views.Controls;
using Xamarin.Forms;

namespace PoMOST.Views.Popups
```

```
{
   public class OtherRequestPopup : BasicPopup
   {
      HelpRequestPopupViewModel vm;
      GradientBlueButton sendButton;

      public OtherRequestPopup()
      {
         BindingContext = vm = ContainerAccessor.Resolve<HelpRequestPopupViewModel>();

         var borderEntry = new BorderEntryControl("Dodatkowe informacje") { HeightRequest = 160 };
         borderEntry.entry.TextChanged += (sender, e) =>
         {
            if (e.NewTextValue == borderEntry.placeholder)
            {
               vm.TextCounter = 0;
               return;
            }
            vm.TextCounter = e.NewTextValue.Length;
         };

         var textCountLabel = new Label { Text = $"Pozostało znaków", Style = CustomStyles.TextPopupStyle, HorizontalTextAlignment = TextAlignment.End, FontSize = 14, Margin = new Thickness(0, -12, 0, 0) };
         textCountLabel.SetBinding(Label.TextProperty, "TextCounter", stringFormat: "Pozostało {0} znaków");

         PopupContent.Content = new StackLayout
         {
            Spacing = 10,
            Children =
            {
               new SvgCachedImage{ WidthRequest=80, HeightRequest=80, Source = SvgImageSource.FromFile("inne"), HorizontalOptions = LayoutOptions.Center, VerticalOptions = LayoutOptions.Center},
```

```
            new Label { Text = "INNE" , Style = CustomStyles.TitlePopupStyle, HorizontalTextAlignment = TextAlignment.Center },
            new Label { Text = "Przykładowy tekst" , Style = CustomStyles.TextPopupStyle },
            borderEntry,
            textCountLabel,
            new StackLayout
            {
               Spacing = 20,
               Orientation = StackOrientation.Horizontal,
               Children =
               {
                  new Button{ Text = "ANULUJ", Command=vm.CloseCommand, Style = CustomStyles.WhiteButtonStyle ,HorizontalOptions = LayoutOptions.FillAndExpand },
                  (sendButton = new GradientBlueButton { Text = "WYŚLIJ", HorizontalOptions = LayoutOptions.FillAndExpand})
               }
            }
         };

         sendButton.Clicked += async (sender, e) =>
         {
            var msg = "potrzebuje pomocy.";
            if (!string.IsNullOrEmpty(borderEntry.CurrentText))
            {
               msg += "\n\n";
               msg += borderEntry.CurrentText;
            }
            await vm.SendHelpRequest(msg, HelpRequest.Categories.OTHER.ToString());
         };
      }
   }
}
```

PhotoPopup:

```
using System;
using PoMOST.Services.Navigation;
using PoMOST.Styles;
using PoMOST.ViewModels.PopupsViewModels;
using Xamarin.Forms;

namespace PoMOST.Views.Popups
{
   public class PhotoPopup : BasicNoBoxPopup
   {
      PhotoPopupViewModel vm;
      Button takePhotoButton;
      Button pickPhotoButton;
      Button cancelButton;

      public PhotoPopup()
      {
         vm = ContainerAccessor.Resolve<PhotoPopupViewModel>();

         var navigationService = ContainerAccessor.Resolve<INavigationService>();

         PopupContent.Content = new StackLayout
         {
            Spacing = 0,
            Children =
            {
               (takePhotoButton = new Button { Text = "ZRÓB ZDJĘCIE", Style = CustomStyles.WhiteButtonStyle , VerticalOptions = LayoutOptions.EndAndExpand, HeightRequest = 60 }),
               (pickPhotoButton =  new Button { Text = "WYBIERZ Z GALERII", Style = CustomStyles.WhiteButtonStyle, Margin = new Thickness(0,5,0,0), HeightRequest = 60 }),
               (cancelButton = new Button { Text = "ANULUJ", Style = CustomStyles.WhiteButtonStyle, Margin = new Thickness(0,20,0,0), HeightRequest = 60 })
            }
```

```
        };

        takePhotoButton.Clicked += async (sender, e) =>
        {
            Value = "TAKE";
            await navigationService.ClosePopup();
        };

        pickPhotoButton.Clicked += async (sender, e) =>
        {
            Value = "PICK";
            await navigationService.ClosePopup();
        };

        cancelButton.Clicked += async(sender, e) =>
        {
            Value = null;
            await navigationService.ClosePopup();
        };
    }
  }
}
```

PostRequestPopup:

```
using System;
using FFImageLoading.Svg.Forms;
using PoMOST.Models;
using PoMOST.Styles;
using PoMOST.ViewModels.PopupsViewModels;
using PoMOST.Views.Controls;
using Xamarin.Forms;

namespace PoMOST.Views.Popups
```

```
{
    public class PostRequestPopup : BasicPopup
    {
        HelpRequestPopupViewModel vm;

        GradientBlueButton sendButton;

        public PostRequestPopup()
        {
            BindingContext = vm = ContainerAccessor.Resolve<HelpRequestPopupViewModel>();

            var borderEntry = new BorderEntryControl("Dodatkowe informacje") { HeightRequest = 160 };
            borderEntry.entry.TextChanged += (sender, e) =>
            {
                if (e.NewTextValue == borderEntry.placeholder)
                {
                    vm.TextCounter = 0;
                    return;
                }
                vm.TextCounter = e.NewTextValue.Length;
            };

            var textCountLabel = new Label { Text = $"Pozostało znaków", Style = CustomStyles.TextPopupStyle, HorizontalTextAlignment = TextAlignment.End, FontSize = 14, Margin = new Thickness(0, -12, 0, 0) };
            textCountLabel.SetBinding(Label.TextProperty, "TextCounter", stringFormat: "Pozostało {0} znaków");

            PopupContent.Content = new StackLayout
            {
                Spacing = 10,
                Children =
                {
```

```
            new SvgCachedImage{ WidthRequest = 80, HeightRequest = 80, Source = SvgImageSource.FromFile("poczta"), HorizontalOptions = LayoutOptions.Center, VerticalOptions = LayoutOptions.Center },
            new Label { Text = "POCZTA" , Style = CustomStyles.TitlePopupStyle, HorizontalTextAlignment = TextAlignment.Center },
            borderEntry,
            textCountLabel,
            new StackLayout
            {
               Spacing = 20,
               Orientation = StackOrientation.Horizontal,
               Children =
               {
                  new Button{ Text = "ANULUJ", Command=vm.CloseCommand, Style = CustomStyles.WhiteButtonStyle ,HorizontalOptions = LayoutOptions.FillAndExpand },
                  (sendButton = new GradientBlueButton { Text = "WYŚLIJ", HorizontalOptions = LayoutOptions.FillAndExpand})
               }
            }
         }
      };

      sendButton.Clicked += async (sender, e) =>
      {
         var msg = "potrzebuje pomocy przy wysłaniu albo odbiorze paczki lub listu.";
         if (!string.IsNullOrEmpty(borderEntry.CurrentText))
         {
            msg += "\n\n";
            msg += borderEntry.CurrentText;
         }
         await vm.SendHelpRequest(msg, HelpRequest.Categories.MAIL.ToString());
      };


   }
}
```

```
}
```

ShopRequestPopup:

```
using System;
using System.Diagnostics;
using FFImageLoading.Svg.Forms;
using PoMOST.Models;
using PoMOST.Styles;
using PoMOST.ViewModels.PopupsViewModels;
using PoMOST.Views.Controls;
using PoMOST.Views.Controls.Radio;
using Xamarin.Forms;

namespace PoMOST.Views.Popups
{
   public class ShopRequestPopup : BasicPopup
   {
      HelpRequestPopupViewModel vm;
      private RadioGroup radioGroup;
      GradientBlueButton sendButton;

      public ShopRequestPopup()
      {
         BindingContext = vm = ContainerAccessor.Resolve<HelpRequestPopupViewModel>();

         var borderEntry = new BorderEntryControl("Dodatkowe informacje") { HeightRequest = 160 };
         borderEntry.entry.TextChanged += (sender, e) =>
         {
            if(e.NewTextValue == borderEntry.placeholder)
            {
               vm.TextCounter = 0;
               return;
```

```
        }
        vm.TextCounter = e.NewTextValue.Length;
    };

    var textCountLabel = new Label { Text = $"Pozostało znaków", Style = CustomStyles.TextPopupStyle, HorizontalTextAlignment = TextAlignment.End, FontSize=14, Margin=new Thickness(0,-12,0,0) };
    textCountLabel.SetBinding(Label.TextProperty, "TextCounter", stringFormat: "Pozostało {0} znaków");

    var radio1 = new RadioButton("Małe zakupy", 0);
    var radio2 = new RadioButton("Większe zakupy", 1);

    radioGroup = new RadioGroup();
    radioGroup.AddRadioButton(radio1);
    radioGroup.AddRadioButton(radio2);
    radioGroup.Select(radio1);

    PopupContent.Content = new StackLayout
    {
        Spacing = 10,
        Children =
        {
            new SvgCachedImage{ WidthRequest=80, HeightRequest=80, Source = SvgImageSource.FromFile("zakupy"), HorizontalOptions = LayoutOptions.Center, VerticalOptions = LayoutOptions.Center },
            new Label { Text = "POMOC W ZAKUPACH" , Style = CustomStyles.TitlePopupStyle, HorizontalTextAlignment = TextAlignment.Center },
            radio1,
            radio2,
            borderEntry,
            textCountLabel,
            new StackLayout
            {
```

```
                Spacing = 20,
                Orientation = StackOrientation.Horizontal,
                Children =
                {
                    new Button{ Text = "ANULUJ", Command=vm.CloseCommand, Style = CustomStyles.WhiteButtonStyle ,HorizontalOptions = LayoutOptions.FillAndExpand },
                    (sendButton = new GradientBlueButton { Text = "WYŚLIJ", HorizontalOptions = LayoutOptions.FillAndExpand})
                }
            }
        }
    };

    sendButton.Clicked+= async (sender, e) =>
    {
        var msg = "";
        if (radioGroup.CurrentSelected.Id == 0)
        {
            msg = "potrzebuje pomocy przy zrobieniu drobnych zakupów.";
        }
        else
        {
            msg = "potrzebuje pomocy przy zrobieniu większych zakupów.";
        }
        Debug.WriteLine("Text "+borderEntry.CurrentText);
        if(!string.IsNullOrEmpty(borderEntry.CurrentText))
        {
            msg += "\n\n";
            msg += borderEntry.CurrentText;
        }
        Debug.WriteLine(msg);
        await vm.SendHelpRequest(msg, HelpRequest.Categories.SHOPPING.ToString());
    };
  }
}
```

```
}
```

TransportRequestPopup:

```
using System;
using FFImageLoading.Svg.Forms;
using PoMOST.Models;
using PoMOST.Styles;
using PoMOST.ViewModels.PopupsViewModels;
using PoMOST.Views.Controls;
using Xamarin.Forms;

namespace PoMOST.Views.Popups
{
   public class TransportRequestPopup : BasicPopup
   {
      HelpRequestPopupViewModel vm;
      GradientBlueButton sendButton;

      public TransportRequestPopup()
      {
         BindingContext = vm = ContainerAccessor.Resolve<HelpRequestPopupViewModel>();

         var borderEntry = new BorderEntryControl("Dodatkowe informacje") { HeightRequest =
160};
         borderEntry.entry.TextChanged += (sender, e) =>
         {
            if (e.NewTextValue == borderEntry.placeholder)
            {
               vm.TextCounter = 0;
               return;
            }
            vm.TextCounter = e.NewTextValue.Length;
         };
```

```
var textCountLabel = new Label { Text = $"Pozostało znaków", Style = CustomStyles.TextPopupStyle, HorizontalTextAlignment = TextAlignment.End, FontSize = 14, Margin = new Thickness(0, -12, 0, 0) };

textCountLabel.SetBinding(Label.TextProperty, "TextCounter", stringFormat: "Pozostało {0} znaków");


PopupContent.Content = new StackLayout
{
   Spacing = 10,
   Children =
   {
      new SvgCachedImage{ WidthRequest=80, HeightRequest=80, Source = SvgImageSource.FromFile("transport"), HorizontalOptions = LayoutOptions.Center, VerticalOptions = LayoutOptions.Center},
      new Label { Text = "POMOC W TRANSPORCIE" , Style = CustomStyles.TitlePopupStyle, HorizontalTextAlignment = TextAlignment.Center},
      borderEntry,
      textCountLabel,
      new StackLayout
      {
         Spacing = 20,
         Orientation = StackOrientation.Horizontal,
         Children =
         {
            new Button{ Text = "ANULUJ", Command=vm.CloseCommand, Style = CustomStyles.WhiteButtonStyle ,HorizontalOptions = LayoutOptions.FillAndExpand },
            (sendButton = new GradientBlueButton { Text = "WYŚLIJ", HorizontalOptions = LayoutOptions.FillAndExpand})
         }
      }
   }
};


sendButton.Clicked += async (sender, e) =>
{
   var msg = "potrzebuje się przetransportować.";
```

```
            if (!string.IsNullOrEmpty(borderEntry.CurrentText))
            {
                msg += "\n\n";
                msg += borderEntry.CurrentText;
            }
            await vm.SendHelpRequest(msg, HelpRequest.Categories.TRANSPORT.ToString());
        };
    }
  }
}
```

VoiceConfirmPopup:

```
using System;
using FFImageLoading.Svg.Forms;
using PoMOST.Models;
using PoMOST.Styles;
using PoMOST.ViewModels.PopupsViewModels;
using PoMOST.Views.Controls;
using Xamarin.Forms;

namespace PoMOST.Views.Popups
{
  public class VoiceConfirmPopup : BasicPopup
  {
    VoiceConfirmPopupViewModel vm;
    GradientBlueButton sendButton;

    public VoiceConfirmPopup(string commandMsg)
    {
      vm = ContainerAccessor.Resolve<VoiceConfirmPopupViewModel>();

      var voiceLabel = new Label()
      {
```

```
            Text = commandMsg.ToUpper(),
            Style = CustomStyles.RegularText
        };

        PopupContent.Content = new StackLayout
        {
            Spacing = 0,
            Children =
            {
                new SvgCachedImage{ WidthRequest=80, HeightRequest=80, Source = SvgImageSource.FromFile("mic_ic"), HorizontalOptions = LayoutOptions.Center, VerticalOptions = LayoutOptions.Center, Margin = new Thickness(0, 5, 0, 0) },
                new Label { Text = "POTWIERDŹ PROŚBĘ" , Style = CustomStyles.TitlePopupStyle, HorizontalTextAlignment = TextAlignment.Center , Margin = new Thickness(0, 15, 0, 0) },
                new StackLayout
                {
                    Margin = new Thickness(0, 21, 0, 0),
                    Spacing = 10,
                    Orientation = StackOrientation.Horizontal,
                    Children =
                    {
                        new Label { Text = "Rozpoznano:", Style= CustomStyles.RegularText},
                        voiceLabel
                    }
                },
                new StackLayout
                {
                    Margin = new Thickness(0, 21, 0, 0),
                    Spacing = 20,
                    Orientation = StackOrientation.Horizontal,
                    Children =
                    {
                        new Button{ Text = "ANULUJ", Style = CustomStyles.WhiteButtonStyle, HorizontalOptions = LayoutOptions.FillAndExpand, Command=vm.CloseCommand },
                        (sendButton = new GradientBlueButton { Text = "WYŚLIJ", HorizontalOptions = LayoutOptions.FillAndExpand })
```

```
            }
        }
    }
};

sendButton.Clicked += async (sender, e) =>
{
    switch (commandMsg)
    {
        case "zakupy":
            await vm.SendHelpRequest("potrzebuje pomocy w zrobieniu zakupów", HelpRequest.Categories.SHOPPING.ToString());
            break;
        case "zdrowie":
            await vm.SendHelpRequest("potrzebuje pomocy w zakupie leków bądź wizyty u lekarza", HelpRequest.Categories.HEALTH.ToString());
            break;
        case "transport":
            await vm.SendHelpRequest("potrzebuje pomocy w przemieszczeniu się”", HelpRequest.Categories.TRANSPORT.ToString());
            break;
        case "poczta":
            await vm.SendHelpRequest("potrzebuje pomocy w w odebraniu lub nadaniu listu lub paczki", HelpRequest.Categories.MAIL.ToString());
            break;
        case "pomocy":
            await vm.SendHelpRequest("jest w niebezpieczeństwie i pilnie potrzebuje pomocy!Skontaktuj się jak najszybciej", HelpRequest.Categories.ALERT.ToString());
            break;
        case "komputer":
            await vm.SendHelpRequest("potrzebuje pomocy w obsłudze komputera lub telefonu", HelpRequest.Categories.IT.ToString());
            break;
        case "inne":
            await vm.SendHelpRequest("potrzebuje pomocy", HelpRequest.Categories.OTHER.ToString());
```

```
                    break;
                }
            };
        }
    }
}
```

SERWER:

Fatowlstudio:

Application.java:

```
package com.fatowlstudio;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.scheduling.annotation.EnableAsync;
import org.springframework.scheduling.annotation.EnableScheduling;
import org.springframework.transaction.annotation.EnableTransactionManagement;

/**
 * Application starting point. It enables all needed Spring Boot auto-configurations and run the app.
 */
@SpringBootApplication
@EnableTransactionManagement
@EnableAsync
@EnableScheduling
public class Application {

    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}
```

Core:

Api

BaseController:

```
package com.fatowlstudio.core.api;

/**
 * Abstract controller class from which other controllers should inherit. This class contains constants for REST paths.
 */
public abstract class BaseController {

    // General api prefix
    private static final String     API_PREFIX =            "/api";

    // Authentication controller
    private static final String     AUTH_PREFIX =           API_PREFIX + "/auth";
    public static final String      LOGIN =                 AUTH_PREFIX + "/login";
    public static final String      LOGOUT =                AUTH_PREFIX + "/logout";

    // Users controller
    public static final String      USERS =                 API_PREFIX + "/users";
    public static final String      USERS_ID =              USERS + "/{userId}";
    public static final String      USERS_LOCATION =        USERS_ID + "/location";
    public static final String      USERS_FCM =             USERS + "/fcm";

    // Help requests controller
    public static final String      HELP_REQUESTS =         API_PREFIX + "/help";
    public static final String      HELP_REQUESTS_OWN =     HELP_REQUESTS + "/own";
    public static final String      HELP_REQUESTS_NEARBY =  HELP_REQUESTS + "/nearby";
```

```
    public static final String HELP_REQUESTS_ID = HELP_REQUESTS +
"/{helpRequestId}";

    // Uploads controller
    public static final String UPLOADS = API_PREFIX + "/uploads";
}
```

HelpRequestController:

```
package com.fatowlstudio.core.api;

import com.fatowlstudio.core.model.dto.HelpRequestDistanceDto;
import com.fatowlstudio.core.model.dto.HelpRequestDto;
import com.fatowlstudio.core.model.dto.NewHelpRequestDto;
import com.fatowlstudio.core.service.HelpRequestService;
import com.fatowlstudio.core.utils.PageRequestBuilder;
import io.swagger.annotations.ApiOperation;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.http.MediaType;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.web.bind.annotation.*;

import javax.validation.Valid;

/**
 * Controller responsible for operations on help requests.
 */
@RestController
public class HelpRequestController extends BaseController {

    private final HelpRequestService helpRequestService;

    /**
```

```
     * Standard constructor.
     *
     * @param helpRequestService autowired helpRequestService
     */
    public HelpRequestController(HelpRequestService helpRequestService) {
        this.helpRequestService = helpRequestService;
    }

    /**
     * API bridge for {@link HelpRequestService#findOwnHelpRequests(Pageable)}. See it for more info.
     */
    @ApiOperation(
        value = "findOwnHelpRequests",
        notes = "Retrieves a page of help requests created by requesting user. By default results are sorted by creation" +
                "date descending."
    )
    @GetMapping(
        value = BaseController.HELP_REQUESTS_OWN,
        produces = MediaType.APPLICATION_JSON_VALUE
    )
    @PreAuthorize("@permissionEvaluator.anyAuthenticated()")
    public Page<HelpRequestDto> findOwnHelpRequests(
        @RequestParam(value = "page", defaultValue = "0,10", required = false) final String page,
        @RequestParam(value = "sort", defaultValue = "creationDate:DESC", required = false) final String sort
    ) {
        return helpRequestService.findOwnHelpRequests(PageRequestBuilder.build(page, sort));
    }

    /**
     * API bridge for {@link HelpRequestService#findNearbyHelpRequests(Pageable)}. See it for more info.
     */
```

```
    @ApiOperation(
        value = "findNearbyHelpRequests",
        notes = "Retrieves a page of help requests with distance. Distance is calculated from location of the user making " +
                "request. Only help requests within that user max distance setting are included in the results. This method " +
                "will skip help requests created by requesting user, to retrieve own help requests please use findOwnHelpRequests " +
                "endpoint. Results are sorted by distance ascending by default."
    )
    @GetMapping(
        value = BaseController.HELP_REQUESTS_NEARBY,
        produces = MediaType.APPLICATION_JSON_VALUE
    )
    @PreAuthorize("@permissionEvaluator.anyAuthenticated()")
    public Page<HelpRequestDistanceDto> findNearbyHelpRequests(
        @RequestParam(value = "page", defaultValue = "0,10", required = false) final String page,
        @RequestParam(value = "sort", defaultValue = "distance:ASC", required = false) final String sort
    ) {
        return helpRequestService.findNearbyHelpRequests(PageRequestBuilder.build(page, sort));
    }

    /**
     * API bridge for {@link HelpRequestService#createHelpRequest(NewHelpRequestDto)}. See it for more info.
     */
    @ApiOperation(
        value = "createHelpRequest",
        notes = "Creates new help request from data passed in the help request form. Requesting user will be automatically " +
                "set as an author of this request. Location and category are obligatory. Category is one of following " +
                "literals: HEALTH, IT, TRANSPORT, MAIL, SHOPPING, ALERT."
    )
    @PostMapping(
```

```
        value = BaseController.HELP_REQUESTS,
        produces = MediaType.APPLICATION_JSON_VALUE,
        consumes = MediaType.APPLICATION_JSON_VALUE
    )
    @PreAuthorize("@permissionEvaluator.anyOfRoles(@authority.NEEDY)")
    public HelpRequestDto createHelpRequest(
        @Valid @RequestBody final NewHelpRequestDto helpRequestForm
    ) {
        return helpRequestService.createHelpRequest(helpRequestForm);
    }

    /**
     * API bridge for {@link HelpRequestService#deleteHelpRequest(long)}. See it for more info.
     */
    @ApiOperation(
        value = "deleteHelpRequest",
        notes = "Removes help request with passed help request id. Request may be removed only if user is logged and he/she " +
                "is author of this request."
    )
    @DeleteMapping(
        value = BaseController.HELP_REQUESTS_ID
    )
    @PreAuthorize("@permissionEvaluator.requestOwner(#helpRequestId)")
    public void deleteHelpRequest(
        @PathVariable final long helpRequestId
    ) {
        helpRequestService.deleteHelpRequest(helpRequestId);
    }
}
```

UserController:

```
package com.fatowlstudio.core.api;
```

```
import com.fatowlstudio.core.model.Location;
import com.fatowlstudio.core.model.dto.UserDetailsDto;
import com.fatowlstudio.core.model.dto.UserRegistrationDto;
import com.fatowlstudio.core.model.dto.UserUpdateDto;
import com.fatowlstudio.core.service.UserDetailsService;
import io.swagger.annotations.ApiOperation;
import org.springframework.http.MediaType;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.web.bind.annotation.*;

import javax.validation.Valid;

/**
 * Controller responsible for operations on users.
 */
@RestController
public class UserController extends BaseController {

    private final UserDetailsService userDetailsService;

    /**
     * Standard constructor.
     *
     * @param userDetailsService autowired userDetailsService
     */
    public UserController(UserDetailsService userDetailsService) {
        this.userDetailsService = userDetailsService;
    }

    /**
     * API bridge for {@link UserDetailsService#findUser(long)}. See it for more info.
     */
    @ApiOperation(
```

```
        value = "findUser",
        notes = "Retrieves single user with passed userId. Regular user can retrieve only himself."
    )
    @GetMapping(
        value = BaseController.USERS_ID,
        produces = MediaType.APPLICATION_JSON_VALUE
    )
    @PreAuthorize("@permissionEvaluator.selfOrAnyOfRoles(#userId, @authority.ADMIN)")
    public UserDetailsDto findUser(
        @PathVariable("userId") final long userId
    ) {
        return userDetailsService.findUser(userId);
    }

    /**
     * API bridge for {@link UserDetailsService#createUser(UserRegistrationDto)}. See it for more info.
     */
    @ApiOperation(
        value = "createUser",
        notes = "Registers user based on provided request body. Created user will be returned in response in case of " +
                "successful processing. <b>gender</b> field is enum being either MALE or FEMALE literal."
    )
    @PostMapping(
        value = BaseController.USERS,
        produces = MediaType.APPLICATION_JSON_VALUE,
        consumes = MediaType.APPLICATION_JSON_VALUE
    )
    public UserDetailsDto createUser(
        @Valid @RequestBody final UserRegistrationDto userDetailsForm
    ) {
        return userDetailsService.createUser(userDetailsForm);
    }
```

```
    /**
     * API bridge for {@link UserDetailsService#updateUser(long, UserUpdateDto)}. See it for more info.
     */
    @ApiOperation(
        value = "updateUser",
        notes = "Updates in the database an user with passed userId."
    )
    @PutMapping(
        value = BaseController.USERS_ID,
        produces = MediaType.APPLICATION_JSON_VALUE,
        consumes = MediaType.APPLICATION_JSON_VALUE
    )
    @PreAuthorize("@permissionEvaluator.self(#userId)")
    public UserDetailsDto updateUser(
        @PathVariable("userId") final long userId,
        @Valid @RequestBody final UserUpdateDto userUpdateForm
    ) {
        return userDetailsService.updateUser(userId, userUpdateForm);
    }

    /**
     * API bridge for {@link UserDetailsService#updateUserLocation(long, Location)}. See it for more info.
     */
    @ApiOperation(
        value = "updateUserLocation",
        notes = "Updates location of the user with passed userId."
    )
    @PutMapping(
        value = BaseController.USERS_LOCATION,
        consumes = MediaType.APPLICATION_JSON_VALUE
    )
    @PreAuthorize("@permissionEvaluator.self(#userId)")
```

```
    public void updateUserLocation(
        @PathVariable("userId") final long userId,
        @RequestBody final Location location
    ) {
        userDetailsService.updateUserLocation(userId, location);
    }

    /**
     * API bridge for {@link UserDetailsService#updateFcmToken(String)}. See it for more info.
     */
    @ApiOperation(
        value = "updateFcmToken",
        notes = "Updates logged user fcm token. If someone already has that token it will be first removed, so one device " +
                "token can be bind with only one account."
    )
    @PutMapping(
        value = BaseController.USERS_FCM
    )
    @PreAuthorize("@permissionEvaluator.anyAuthenticated()")
    public void updateFcmToken(
        @RequestParam("fcmToken") String fcmToken
    ) {
        userDetailsService.updateFcmToken(fcmToken);
    }
}
```

Config:

BeanConfig

```
package com.fatowlstudio.core.config;

import com.fasterxml.jackson.databind.ObjectMapper;
```

```
import com.fasterxml.jackson.datatype.jsr310.JavaTimeModule;
import com.fatowlstudio.core.service.FcmNotificationService;
import de.bytefish.fcmjava.client.FcmClient;
import de.bytefish.fcmjava.client.settings.PropertiesBasedSettings;
import de.bytefish.fcmjava.http.client.IFcmClient;
import org.apache.log4j.Logger;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Primary;

import java.io.IOException;
import java.time.ZonedDateTime;
import java.util.Properties;

/**
 * Configuration for general beans used in application.
 */
@Configuration
public class BeanConfig {

    Logger logger = Logger.getLogger(this.getClass());

    @Value("${fcm.api.url}")
    private String apiUrl;

    @Value("${fcm.api.key}")
    private String apiKey;

    /**
     * Creates Google's Firebase Cloud Messaging client for use in {@link FcmNotificationService}
to send push messages.
     * Client configuration has to be provided in application.properties file as 'fcm.api.url' and
'fcm.api.key' properties.
     *
```

```
     * @return ready to use configured fcm client
     */
    @Bean
    public IFcmClient iFcmClient() throws IOException {

        Properties properties = new Properties();
        properties.put("fcm.api.url", apiUrl);
        properties.put("fcm.api.key", apiKey);

        PropertiesBasedSettings propertiesBasedSettings = PropertiesBasedSettings
                .createFromProperties(properties);

        return new FcmClient(propertiesBasedSettings);
    }

    /**
     * JSR-310 custom mappers bean for appropriate {@link ZonedDateTime} class mapping, calculating and formatting it
     * to/from UTC timezone when coming forth and back.
     *
     * @return custom object mapper bean for jackson module
     */
    @Bean
    @Primary
    public ObjectMapper zonedDateTimeObjectMapper() {
        ObjectMapper objectMapper = new ObjectMapper();
        JavaTimeModule javaTimeModule = new JavaTimeModule();
        javaTimeModule.addSerializer(ZonedDateTime.class, new CustomSerializer.CustomZonedDateTimeSerializer());
        javaTimeModule.addDeserializer(ZonedDateTime.class, new CustomSerializer.CustomZonedDateTimeDeserializer());
        objectMapper.registerModule(javaTimeModule);
        return objectMapper;
    }
}
```

CustomSerializer

```
package com.fatowlstudio.core.config;

import com.fasterxml.jackson.core.JsonGenerator;
import com.fasterxml.jackson.core.JsonParser;
import com.fasterxml.jackson.databind.DeserializationContext;
import com.fasterxml.jackson.databind.JsonDeserializer;
import com.fasterxml.jackson.databind.JsonSerializer;
import com.fasterxml.jackson.databind.SerializerProvider;

import java.io.IOException;
import java.time.ZoneOffset;
import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;

import static java.time.format.DateTimeFormatter.ofPattern;

/**
 * Class containing custom serializers and deserializers for jackson.
 */
abstract class CustomSerializer {

   private static final DateTimeFormatter FORMATTER = ofPattern("yyyy-MM-dd'T'HH:mm:ss'Z'");

   public static class CustomZonedDateTimeSerializer extends JsonSerializer<ZonedDateTime> {
      @Override
      public void serialize(ZonedDateTime val, JsonGenerator gen, SerializerProvider provider) throws IOException {
         ZonedDateTime valInUtc = val.withZoneSameInstant(ZoneOffset.UTC);
         gen.writeString(valInUtc.format(FORMATTER));
      }
   }
```

```
    public static class CustomZonedDateTimeDeserializer extends JsonDeserializer<ZonedDateTime> {
        @Override
        public ZonedDateTime deserialize(JsonParser parser, DeserializationContext ctxt) throws IOException {
            return ZonedDateTime.parse(parser.getValueAsString(), FORMATTER).withZoneSameInstant(ZoneOffset.UTC);
        }
    }
}
```

SwaggerConfig

```
package com.fatowlstudio.core.config;

import com.fatowlstudio.security.config.WebSecurityConfig;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.bind.annotation.RestController;
import springfox.documentation.builders.ParameterBuilder;
import springfox.documentation.builders.PathSelectors;
import springfox.documentation.builders.RequestHandlerSelectors;
import springfox.documentation.schema.ModelRef;
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spring.web.plugins.Docket;
import springfox.documentation.swagger2.annotations.EnableSwagger2;

import java.time.ZonedDateTime;
import java.util.Collections;

/**
 * Swagger configuration. It will pick up any Spring @RestController and all of its actions
 * and automatically set it up to use for use by swagger ui. Springfox implementation is used.
 */
```

```
@Configuration
@EnableSwagger2
public class SwaggerConfig {

    /**
     * Creates standard Springfox bean for swagger.
     *
     * @return Springfox Docket
     */
    @Bean
    public Docket api() {
        return new Docket(DocumentationType.SWAGGER_2)
                .directModelSubstitute(ZonedDateTime.class, String.class)
                .select()
                  .apis(RequestHandlerSelectors.withClassAnnotation(RestController.class))
                  .paths(PathSelectors.any())
                .build()
                  .globalOperationParameters(
                    Collections.singletonList(
                      new ParameterBuilder()
                        .name(WebSecurityConfig.JWT_TOKEN_HEADER)
                        .description("Authorization header")
                        .modelRef(new ModelRef("string"))
                        .parameterType("header")
                        .required(false)
                        .build()
                    )
                  );
    }
}
```

Ex

LoginNotAvailableException:

```
package com.fatowlstudio.core.ex;

import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.ResponseStatus;

/**
 * Exception thrown when user tries to register with use of non-available login.
 */
@ResponseStatus(value = HttpStatus.CONFLICT, reason = "Login not available")
public class LoginNotAvailableException extends RuntimeException {
}
```

Mapper

HelpRequestMapper:

```
package com.fatowlstudio.core.mapper;

import com.fatowlstudio.core.model.HelpRequest;
import com.fatowlstudio.core.model.Location;
import com.fatowlstudio.core.model.dto.HelpRequestDistanceDto;
import com.fatowlstudio.core.model.dto.HelpRequestDto;

/**
 * Class containing mapping logic for {@link HelpRequest} entity.
 */
public abstract class HelpRequestMapper {

    /**
     * Method mapping {@link HelpRequest} entity to it's general purpose dto.
     *
     * @param entity entity to be mapped
     * @return dto of that entity
```

```
     */
    public static HelpRequestDto toDto(HelpRequest entity) {
        HelpRequestDto dto = new HelpRequestDto();
        dto.setAuthor(UserDetailsMapper.toDto(entity.getAuthor()));
        dto.setCategory(entity.getCategory());
        dto.setCreationDate(entity.getCreationDate());
        dto.setId(entity.getId());
        dto.setLocation(entity.getLocation());
        dto.setText(entity.getText());
        return dto;
    }

    /**
     * Method mapping {@link HelpRequest} entity to it's general puropose dto extended with distance calculated from
     * passed longitude and latitude.
     *
     * @param entity entity to be mapped
     * @return dto of that entity with calculated distance
     */
    public static HelpRequestDistanceDto toDistanceDto(HelpRequest entity, double userLatitude, double userLongitude) {
        HelpRequestDistanceDto dto = new HelpRequestDistanceDto();
        dto.setAuthor(UserDetailsMapper.toDto(entity.getAuthor()));
        dto.setCategory(entity.getCategory());
        dto.setCreationDate(entity.getCreationDate());
        dto.setId(entity.getId());
        dto.setLocation(entity.getLocation());
        dto.setText(entity.getText());
        dto.setDistance(calculateDistanceFromPoint(entity.getLocation(), userLatitude, userLongitude));
        return dto;
    }

    /**
```

```
    * Calculates distance in meters between entity position and user position. Entity position should be passed as
    * a JPA location entity and user coordinates are passed as regular arguments.
    *
    * @param entityPosition position of the help request entity
    * @param userLatitude latitude of the user for which distance is being calculated
    * @param userLongitude longitude of the user for which distance is being calculated
    * @return distance in kilometers between those two points
    */
   private static int calculateDistanceFromPoint(Location entityPosition, double userLatitude, double userLongitude) {
       float distanceInKm = 6371.0f * (float) Math.acos(
          Math.cos(Math.toRadians(userLatitude))
             * Math.cos(Math.toRadians(entityPosition.getLatitude()))
             * Math.cos(Math.toRadians(entityPosition.getLongitude()) - Math.toRadians(userLongitude))
             + Math.sin(Math.toRadians(userLatitude))
             * Math.sin(Math.toRadians(entityPosition.getLatitude()))
       );

       // Return distance in meters
       return Math.round(distanceInKm * 1000f);
   }
}
```

UserDetailsMapper:

```
package com.fatowlstudio.core.mapper;

import com.fatowlstudio.core.model.UserDetails;
import com.fatowlstudio.core.model.dto.UserDetailsDto;
import com.fatowlstudio.security.model.Authority;

/**
 * Class containing mapping logic for {@link UserDetails} entity.
```

```
 */
public abstract class UserDetailsMapper {

    /**
     * Method mapping {@link UserDetails} entity to it's general purpose dto.
     *
     * @param entity entity to be mapped
     * @return dto of that entity
     */
    public static UserDetailsDto toDto(UserDetails entity) {
        UserDetailsDto dto = new UserDetailsDto();
        dto.setId(entity.getId());
        dto.setPhone(entity.getPhone());
        dto.setName(entity.getName());
        dto.setGender(entity.getGender());
        dto.setAvatar(entity.getAvatar());
        dto.setNeedy(entity.getAuthorities().contains(Authority.NEEDY));
        dto.setHelper(entity.getAuthorities().contains(Authority.HELPER));
        dto.setDescription(entity.getDescription());
        dto.setNotificationRadius(entity.getNotificationRadius());
        dto.setNotificationsActive(entity.isNotificationsActive());
        dto.setLocation(entity.getLocation());
        return dto;
    }
}
```

Model

Dto

HelpRequest

```
package com.fatowlstudio.core.model;
```

```
import com.fatowlstudio.core.model.value.HelpRequestCategory;

import javax.persistence.*;
import java.time.ZonedDateTime;

/**
 * Model for storing help requests in the database.
 */
@Entity
public class HelpRequest {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;

    @Column(nullable = false)
    @Enumerated(EnumType.STRING)
    private HelpRequestCategory category;

    @Column(length = 1000)
    private String text;

    @Embedded
    private Location location;

    @Column(nullable = false)
    private ZonedDateTime creationDate;

    @ManyToOne(optional = false, fetch = FetchType.LAZY)
    @JoinColumn(nullable = false, name = "author_id")
    private UserDetails author;

    /**
     * Standard constructor.
```

```
 */
public HelpRequest() {
}

/*
 * GETTERS AND SETTERS
 */

public long getId() {
    return id;
}

public void setId(long id) {
    this.id = id;
}

public HelpRequestCategory getCategory() {
    return category;
}

public void setCategory(HelpRequestCategory category) {
    this.category = category;
}

public String getText() {
    return text;
}

public void setText(String text) {
    this.text = text;
}

public Location getLocation() {
    return location;
```

```
    }

    public void setLocation(Location location) {
        this.location = location;
    }

    public ZonedDateTime getCreationDate() {
        return creationDate;
    }

    public void setCreationDate(ZonedDateTime creationDate) {
        this.creationDate = creationDate;
    }

    public UserDetails getAuthor() {
        return author;
    }

    public void setAuthor(UserDetails author) {
        this.author = author;
    }
}
```

Location

```
package com.fatowlstudio.core.model;

import javax.persistence.Column;
import javax.persistence.Embeddable;
import javax.validation.constraints.NotNull;

/**
 * Embedded class containing coordinates for help requests.
 */
```

```
@Embeddable
public class Location {

    @NotNull
    @Column(nullable = false)
    private double latitude;

    @NotNull
    @Column(nullable = false)
    private double longitude;

    public Location() {
    }

    public double getLatitude() {
        return latitude;
    }

    public void setLatitude(double latitude) {
        this.latitude = latitude;
    }

    public double getLongitude() {
        return longitude;
    }

    public void setLongitude(double longitude) {
        this.longitude = longitude;
    }
}
```

UserDetails

```
package com.fatowlstudio.core.model;
```

```
import com.fatowlstudio.core.model.value.Gender;

import javax.persistence.*;
import java.util.List;

/**
 * Model for storing users details in the database.
 */
@Entity
public class UserDetails {

    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;

    @Column(unique = true, nullable = false)
    private String phone;

    @Column(nullable = false)
    private String name;

    @Column(nullable = false)
    private String authorities;

    @Enumerated(EnumType.STRING)
    @Column(nullable = false)
    private Gender gender;

    private String avatar;

    @Column(columnDefinition = "text")
    private String description;

    @Column(nullable = false)
```

```
private int notificationRadius;

@Column(nullable = false)
private boolean notificationsActive;

@Embedded
private Location location;

@Column(columnDefinition = "text")
private String fcmToken;

/**
 * RELATIONAL FIELDS
 */

@OneToMany(mappedBy = "author", cascade = CascadeType.REMOVE)
private List<HelpRequest> helpRequests;

/**
 * Standard constructor.
 */
public UserDetails() {
}

/*
 * GETTERS AND SETTERS
 */

public long getId() {
    return id;
}

public void setId(long id) {
    this.id = id;
```

```
}

public String getPhone() {
    return phone;
}

public void setPhone(String phone) {
    this.phone = phone;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public String getAuthorities() {
    return authorities;
}

public void setAuthorities(String authorities) {
    this.authorities = authorities;
}

public Gender getGender() {
    return gender;
}

public void setGender(Gender gender) {
    this.gender = gender;
}
```

```
public String getAvatar() {
    return avatar;
}

public void setAvatar(String avatar) {
    this.avatar = avatar;
}

public String getDescription() {
    return description;
}

public void setDescription(String description) {
    this.description = description;
}

public int getNotificationRadius() {
    return notificationRadius;
}

public void setNotificationRadius(int notificationRadius) {
    this.notificationRadius = notificationRadius;
}

public boolean isNotificationsActive() {
    return notificationsActive;
}

public void setNotificationsActive(boolean notificationsActive) {
    this.notificationsActive = notificationsActive;
}

public List<HelpRequest> getHelpRequests() {
    return helpRequests;
```

```
    }

    public void setHelpRequests(List<HelpRequest> helpRequests) {
        this.helpRequests = helpRequests;
    }

    public Location getLocation() {
        return location;
    }

    public void setLocation(Location location) {
        this.location = location;
    }

    public String getFcmToken() {
        return fcmToken;
    }

    public void setFcmToken(String fcmToken) {
        this.fcmToken = fcmToken;
    }
}
```

Value

Gender

```
package com.fatowlstudio.core.model.value;

/**
 * Enum class representing possible genders.
 */
public enum Gender {
```

```
    MALE, FEMALE
}
```

HelpRequestCategory:

```
package com.fatowlstudio.core.model.value;

/**
 * Enum class representing possible categories of help request.
 */
public enum HelpRequestCategory {

    HEALTH, IT, TRANSPORT, MAIL, SHOPPING, ALERT, OTHER
}
```

Wrapper

```
package com.fatowlstudio.core.model.wrapper;

/**
 * Simple primitive type wrapper, used to wrap primitive types when they are only one thing to return from controller,
 * so they can be used as a regular JSON object.
 */
public class Wrapper<T> {

    private T value;

    public Wrapper(T value) {
        this.value = value;
    }

    public T getValue() {
        return value;
```

```
    }

    public void setValue(T value) {
        this.value = value;
    }
}
```

Repository

HelpRequestRepository

```
package com.fatowlstudio.core.repository;

import com.fatowlstudio.core.model.HelpRequest;
import com.fatowlstudio.core.model.dto.HelpRequestDto;
import com.fatowlstudio.upload.model.UploadModel;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;

import java.time.ZonedDateTime;
import java.util.List;

/**
 * Repository responsible for operations on {@link HelpRequest}.
 */
public interface HelpRequestRepository extends JpaRepository<HelpRequest, Long> {

    /**
     * Retrieves id of the user who is author of request with passed id.
     *
     * @param requestId id of the request
```

```
     * @return id of the user who made this request
     */
    @Query("SELECT hr.author.id FROM HelpRequest hr WHERE hr.id = :requestId")
    long finOwnerIdOfHelpRequest(@Param("requestId") long requestId);

    /**
     * Retrieves a page of help requests created by user with passed userId.
     *
     * @param userId id of the user of which help requests should be retrieved
     * @param pageRequest pageable object from which page request will be built
     * @return page of help requests created by requesting user
     */
    @Query("SELECT hr FROM HelpRequest hr WHERE hr.author.id = :userId")
    Page<HelpRequest> findHelpRequestsCreatedBy(@Param("userId") long userId, Pageable pageRequest);

    /**
     * Retrieves list of old help requests. Help request is considered as unused if it was created before a date passed
     * as a parameter.
     *
     * @return list of old help requests
     */
    @Query("SELECT hp FROM HelpRequest hp WHERE hp.creationDate < :limit")
    List<HelpRequest> findOldHelpRequests(@Param("limit") ZonedDateTime limit);

    /**
     * Retrieves a page of help requests with distance. Distance is calculated from latitude and longitude passed as an
     * arguments. Only help requests within passed max distance are included in the results. This method will skip help
     * requests created by user with passed userId.
     *
     * This is native query as jpql does not support required mySql math functions. All distances are taken and returned
     * in kilometers.
```

```
 *
 * @param pageable pageable object from which page request will be built
 * @param userId id of the user for which distances are calculated
 * @param userLatitude actual latitude of the user with user id
 * @param userLongitude actual longitude of the user with user id
 * @param maxDistance max distance in which we should look for help requests
 * @return page of help requests within max distance from requested point
 */
@Query(
    countQuery =
    "SELECT count(*), ( " +
        "6371 * acos (" +
            "cos ( radians (:userLatitude) ) " +
            "* cos ( radians (latitude) ) " +
            "* cos ( radians (longitude) - radians (:userLongitude) ) " +
            "+ sin ( radians (:userLatitude) ) " +
            "* sin (radians (latitude) ) " +
        ")" +
    ") AS distance " +
    "FROM help_request " +
    "WHERE author_id <> :userId " +
    "HAVING distance < :maxDistance ",

    value =
    "SELECT *, ( " +
        "6371 * acos (" +
            "cos ( radians (:userLatitude) ) " +
            "* cos ( radians (latitude) ) " +
            "* cos ( radians (longitude) - radians (:userLongitude) ) " +
            "+ sin ( radians (:userLatitude) ) " +
            "* sin (radians (latitude) ) " +
        ")" +
    ") AS distance " +
    "FROM help_request " +
```

```
        "WHERE author_id <> :userId " +
        "HAVING distance < :maxDistance " +
        "ORDER BY ?#{#pageable}",

        nativeQuery = true
    )
    Page<HelpRequest> findNearbyHelpRequests(
        @Param("userId") long userId,
        @Param("userLatitude") double userLatitude,
        @Param("userLongitude") double userLongitude,
        @Param("maxDistance") double maxDistance,
        Pageable pageable);
}
```

UserDetailsRepository

```
package com.fatowlstudio.core.repository;

import com.fatowlstudio.core.model.HelpRequest;
import com.fatowlstudio.core.model.UserDetails;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Modifying;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;

import java.util.List;

/**
 * Repository responsible for operations on {@link UserDetails}.
 */
public interface UserDetailsRepository extends JpaRepository<UserDetails, Long> {
```

```
    /**
     * Finds user by phone.
     *
     * @param phone phone of user we search for
     * @return user with provided email
     */
    @Query("SELECT ud FROM UserDetails ud WHERE ud.phone = :phone")
    UserDetails findByPhone(@Param("phone") String phone);


    /**
     * Checks if passed phone number is available as a login for a user with passed user id. Will return true if so or
     * false otherwise.
     *
     * @param userId id of the user for which phone should be checked
     * @param phone phone to be checked
     * @return true is phone is available for this user, false otherwise
     */
    @Query("SELECT (NOT (COUNT(ud) > 0)) FROM UserDetails ud WHERE ud.phone = :phone AND ud.id <> :userId")
    boolean isPhoneAvailableForUser(@Param("userId") long userId, @Param("phone") String phone);


    /**
     * Removes all fcmTokens from user profiles whit it being equal to passed fcmToken.
     *
     * @param fcmToken fcmToken to be removed
     */
    @Modifying
    @Query("UPDATE UserDetails ud SET ud.fcmToken = null WHERE ud.fcmToken = :fcmToken")
    void removeFcmToken(@Param("fcmToken") String fcmToken);


    /**
     * Updates chosen user fcmToken.
```

```
     *
     * @param userId id of the user whose fcmToken should be updated
     * @param fcmToken fcm token to be set
     */
    @Modifying
    @Query("UPDATE UserDetails ud SET ud.fcmToken = :fcmToken WHERE ud.id = :userId")
    void updateFcmToken(@Param("userId") long userId, @Param("fcmToken") String fcmToken);


    /**
     * Retrieves a list of helper users (with notifications active) nearby passed help request location coordinates. By
     * nearby is meant that help request is inside helper notification radius defined in his/her profile. ExcludedUserId
     * has to be passed to exclude from search user who created this help request.
     *
     * This is native query as jpql does not support required mySql math functions.
     *
     * @param excludedUserId id of the user to exclude from search
     * @param helpRequestLatitude actual latitude of the help request
     * @param helpRequestLongitude  actual longitude of the help request
     * @return list of helpers with this help request within their notification radius
     */
    @Query(
        value =
            "SELECT *, ( " +
                "6371 * acos (" +
                "cos ( radians (:helpRequestLatitude) ) " +
                "* cos ( radians (latitude) ) " +
                "* cos ( radians (longitude) - radians (:helpRequestLongitude) ) " +
                "+ sin ( radians (:helpRequestLatitude) ) " +
                "* sin (radians (latitude) ) " +
            ")" +
        ") AS distance " +
        "FROM user_details " +
```

```
        "WHERE id <> :excludedUserId AND notifications_active = true AND fcm_token IS NOT NULL " +
        "HAVING distance < (notification_radius/1000) ",

        nativeQuery = true
    )
    List<UserDetails> findNearbyHelpers(
        @Param("excludedUserId") long excludedUserId,
        @Param("helpRequestLatitude") double helpRequestLatitude,
        @Param("helpRequestLongitude") double helpRequestLongitude);
}
```

Service

AccountService:

```
package com.fatowlstudio.core.service;

/**
 * Service responsible for more specific account related actions, as password managing etc.
 */
public interface AccountService {

    /**
     * Returns role name considered as the most basic for any new user.
     *
     * @return new user base authority
     */
    String retrieveBaseAuthority();

    /**
     * Returns role names generated basing on requested user roles.
     *
     * @param helper defines if user is helper
```

```
     * @param needy defines if user is needy
     * @return user authorities
     */
    String retrieveAuthorities(boolean helper, boolean needy);

    /**
     * Checks if passed phone number is available as a login for a user with passed user id. Will return true if so or
     * false otherwise.
     *
     * @param userId id of the user for which phone should be checked
     * @param phone phone to be checked
     * @return true is phone is available for this user, false otherwise
     */
    boolean isPhoneAvailableForUser(long userId, String phone);
}
```

FcmNotificationService

```
package com.fatowlstudio.core.service;

/**
 * Service responsible for more specific account related actions, as password managing etc.
 */
public interface AccountService {

    /**
     * Returns role name considered as the most basic for any new user.
     *
     * @return new user base authority
     */
    String retrieveBaseAuthority();

    /**
```

```
    * Returns role names generated basing on requested user roles.
    *
    * @param helper defines if user is helper
    * @param needy defines if user is needy
    * @return user authorities
    */
   String retrieveAuthorities(boolean helper, boolean needy);

   /**
    * Checks if passed phone number is available as a login for a user with passed user id. Will return true if so or
    * false otherwise.
    *
    * @param userId id of the user for which phone should be checked
    * @param phone phone to be checked
    * @return true is phone is available for this user, false otherwise
    */
   boolean isPhoneAvailableForUser(long userId, String phone);
}
```

HelpRequestService

```
package com.fatowlstudio.core.service;

import com.fatowlstudio.core.model.HelpRequest;
import com.fatowlstudio.core.model.dto.HelpRequestDistanceDto;
import com.fatowlstudio.core.model.dto.HelpRequestDto;
import com.fatowlstudio.core.model.dto.NewHelpRequestDto;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;

import java.util.List;

/**
```

```
 * Service responsible for operations on help requests.
 */
public interface HelpRequestService {

    /**
     * Retrieves a page of help requests created by requesting user.
     *
     * @param pageRequest pageable object from which page request will be built
     * @return page of help requests created by requesting user
     */
    Page<HelpRequestDto> findOwnHelpRequests(Pageable pageRequest);

    /**
     * Retrieves a page of help requests with distance. Distance is calculated from location of the user making request.
     * Only help requests within that user max distance setting are included in the results. This method will skip help
     * requests created by requesting user, to retrieve own help requests please use
     * {@link HelpRequestService#findOwnHelpRequests(Pageable)}.
     *
     * @param pageRequest pageable object from which page request will be built
     * @return page of help requests with calculated distance from requesting user
     */
    Page<HelpRequestDistanceDto> findNearbyHelpRequests(Pageable pageRequest);

    /**
     * Creates new help request from data passed in the help request form. Requesting user will be automatically set
     * as an author of this request.
     *
     * @param helpRequestForm form with data of new help request
     * @return representation of created help request
     */
    HelpRequestDto createHelpRequest(final NewHelpRequestDto helpRequestForm);
```

```
    /**
     * Removes help request with passed help request id.
     *
     * @param helpRequestId id of the help request to remove
     */
    void deleteHelpRequest(final long helpRequestId);

    /**
     * Removes passed help request.
     *
     * @param helpRequest help request to remove
     */
    void deleteHelpRequest(final HelpRequest helpRequest);

    /**
     * Returns list of old help requests. Help request is considered old if its older that two days.
     *
     * @return list of old help requests
     */
    List<HelpRequest> findOldHelpRequests();
}
```

Impl

AccountServiceImpl

```
package com.fatowlstudio.core.service.impl;

import com.fatowlstudio.core.repository.UserDetailsRepository;
import com.fatowlstudio.core.service.AccountService;
import com.fatowlstudio.core.utils.CodingUtils;
import com.fatowlstudio.security.config.WebSecurityConfig;
import com.fatowlstudio.security.model.Authority;
import org.apache.log4j.Logger;
```

```
import org.springframework.security.crypto.bcrypt.BCrypt;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

/**
 * Default implementation for {@link AccountService} interface. See the interface for public
 * API documentation.
 */
@Service
@Transactional
public class AccountServiceImpl implements AccountService {

    private final Logger LOGGER = Logger.getLogger(this.getClass());

    private final UserDetailsRepository userDetailsRepository;

    public AccountServiceImpl(UserDetailsRepository userDetailsRepository) {
        this.userDetailsRepository = userDetailsRepository;
    }

    @Override
    public String retrieveBaseAuthority() {
        return Authority.USER;
    }

    @Override
    public String retrieveAuthorities(boolean helper, boolean needy) {
        String authorities = retrieveBaseAuthority();
        if (helper) authorities = authorities.concat(", ").concat(Authority.HELPER);
        if (needy) authorities = authorities.concat(", ").concat(Authority.NEEDY);
        return authorities;
    }

    @Override
```

```
    public boolean isPhoneAvailableForUser(long userId, String phone) {
        return userDetailsRepository.isPhoneAvailableForUser(userId, phone);
    }
}
```

FcmNotificationServiceImpl

```
package com.fatowlstudio.core.service.impl;

import com.fatowlstudio.core.model.Location;
import com.fatowlstudio.core.model.dto.NotificationPayloadDto;
import com.fatowlstudio.core.repository.UserDetailsRepository;
import com.fatowlstudio.core.service.FcmNotificationService;
import de.bytefish.fcmjava.http.client.IFcmClient;
import de.bytefish.fcmjava.model.options.FcmMessageOptions;
import de.bytefish.fcmjava.requests.data.DataUnicastMessage;
import de.bytefish.fcmjava.requests.notification.NotificationPayload;
import de.bytefish.fcmjava.responses.FcmMessageResponse;
import org.apache.log4j.Logger;
import org.springframework.scheduling.annotation.Async;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import java.util.Collections;

/**
 * Default implementation for {@link FcmNotificationService} interface. See the interface for public API documentation.
 */
@Service
@Transactional
public class FcmNotificationServiceImpl implements FcmNotificationService {

    private final Logger LOGGER = Logger.getLogger(this.getClass());
```

```
    private final IFcmClient fcmClient;
    private final UserDetailsRepository userDetailsRepository;

    public FcmNotificationServiceImpl(IFcmClient fcmClient, UserDetailsRepository userDetailsRepository) {
        this.fcmClient = fcmClient;
        this.userDetailsRepository = userDetailsRepository;
    }

    @Override
    @Async
    public void notifyNearbyHelpers(Location helpRequestLocation, long helpRequestId, long excludedUserId) {
        userDetailsRepository.findNearbyHelpers(
            excludedUserId,
            helpRequestLocation.getLatitude(),
            helpRequestLocation.getLongitude()
        ).forEach(userToNotify -> {
            sendNotification(
                userToNotify.getFcmToken(),
                Collections.singletonMap("NEW_HELP_REQUEST_ID", helpRequestId),
                new NotificationPayloadDto(
                    "Nowa prośba o pomoc",
                    "W Twojej okolicy pojawiła się nowa prośba o pomoc")
            );
        });
    }

    /**
     * Sends firebase push notification to the android/ios mobile client.
     *
     * @param fcmToken users fcm api registration key to which message will be sent
     * @param data custom data to sent alongside standard notification payload
     * @param payloadDto payload dto object from which fcm payload will be constructed
     */
```

```
    private void sendNotification(String fcmToken, Object data, NotificationPayloadDto payloadDto) {

        FcmMessageOptions options = FcmMessageOptions.builder().build();
        NotificationPayload payload = NotificationPayload.builder()
            .setBody(payloadDto.getBody())
            .setTitle(payloadDto.getTitle())
            .setBadge("1")
            .setSound("default")
            .build();

        DataUnicastMessage notification = new DataUnicastMessage(options, fcmToken, data, payload);
        FcmMessageResponse notificationResponse = fcmClient.send(notification);

        notificationResponse.getResults().forEach(
            result -> {
                if (result.getErrorCode() != null)
                    LOGGER.warn("Failed to sent notification to " + fcmToken);
            }
        );
    }
}
```

HelpRequestServiceImpl

```
package com.fatowlstudio.core.service.impl;

import com.fatowlstudio.core.mapper.HelpRequestMapper;
import com.fatowlstudio.core.model.HelpRequest;
import com.fatowlstudio.core.model.UserDetails;
import com.fatowlstudio.core.model.dto.HelpRequestDistanceDto;
import com.fatowlstudio.core.model.dto.HelpRequestDto;
import com.fatowlstudio.core.model.dto.NewHelpRequestDto;
import com.fatowlstudio.core.repository.HelpRequestRepository;
```

```
import com.fatowlstudio.core.service.FcmNotificationService;
import com.fatowlstudio.core.service.HelpRequestService;
import com.fatowlstudio.security.service.AuthService;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import java.time.ZoneOffset;
import java.time.ZonedDateTime;
import java.time.temporal.ChronoUnit;
import java.util.List;

/**
 * Default implementation for {@link HelpRequestService} interface. See the interface for public API documentation.
 */
@Service
@Transactional
public class HelpRequestServiceImpl implements HelpRequestService {

    @PersistenceContext
    private EntityManager entityManager;

    private final HelpRequestRepository helpRequestRepository;
    private final AuthService authService;
    private final FcmNotificationService fcmNotificationService;

    public HelpRequestServiceImpl(HelpRequestRepository helpRequestRepository, AuthService authService,
            FcmNotificationService fcmNotificationService) {
        this.helpRequestRepository = helpRequestRepository;
        this.authService = authService;
```

```
        this.fcmNotificationService = fcmNotificationService;
    }

    @Override
    public Page<HelpRequestDto> findOwnHelpRequests(Pageable pageRequest) {
        return
helpRequestRepository.findHelpRequestsCreatedBy(authService.getLoggedUserPrincipal().getUser
Id(), pageRequest)
                .map(HelpRequestMapper::toDto);
    }

    @Override
    public Page<HelpRequestDistanceDto> findNearbyHelpRequests(Pageable pageRequest) {
        UserDetails requestingUser = authService.getLoggedUser();
        return helpRequestRepository.findNearbyHelpRequests(
                requestingUser.getId(),
                requestingUser.getLocation().getLatitude(),
                requestingUser.getLocation().getLongitude(),
                requestingUser.getNotificationRadius() / 1000d,
                pageRequest
        ).map(helpRequest -> HelpRequestMapper.toDistanceDto(
                helpRequest,                                    requestingUser.getLocation().getLatitude(),
requestingUser.getLocation().getLongitude()));
    }

    @Override
    public HelpRequestDto createHelpRequest(final NewHelpRequestDto helpRequestForm) {
        long creatorId = authService.getLoggedUserPrincipal().getUserId();
        HelpRequest newHelpRequest = new HelpRequest();
        newHelpRequest.setAuthor(entityManager.getReference(UserDetails.class, creatorId));
        newHelpRequest.setCategory(helpRequestForm.getCategory());
        newHelpRequest.setCreationDate(ZonedDateTime.now(ZoneOffset.UTC));
        newHelpRequest.setLocation(helpRequestForm.getLocation());
        newHelpRequest.setText(helpRequestForm.getText());
```

```
        HelpRequestDto requestDto = HelpRequestMapper.toDto(helpRequestRepository.save(newHelpRequest));

        fcmNotificationService.notifyNearbyHelpers(newHelpRequest.getLocation(), newHelpRequest.getId(), creatorId);
        return requestDto;
    }

    @Override
    public void deleteHelpRequest(final long helpRequestId) {
        helpRequestRepository.delete(helpRequestId);
    }

    @Override
    public void deleteHelpRequest(HelpRequest helpRequest) {
        deleteHelpRequest(helpRequest.getId());
    }

    @Override
    public List<HelpRequest> findOldHelpRequests() {
        return helpRequestRepository.findOldHelpRequests(ZonedDateTime.now(ZoneOffset.UTC).minus(2L, ChronoUnit.DAYS));
    }
}
```

UserDetailsServiceImpl

```
package com.fatowlstudio.core.service.impl;

import com.fatowlstudio.core.ex.LoginNotAvailableException;
import com.fatowlstudio.core.mapper.UserDetailsMapper;
import com.fatowlstudio.core.model.Location;
import com.fatowlstudio.core.model.UserDetails;
import com.fatowlstudio.core.model.dto.UserDetailsDto;
```

```
import com.fatowlstudio.core.model.dto.UserRegistrationDto;
import com.fatowlstudio.core.model.dto.UserUpdateDto;
import com.fatowlstudio.core.repository.UserDetailsRepository;
import com.fatowlstudio.core.service.AccountService;
import com.fatowlstudio.core.service.UserDetailsService;
import com.fatowlstudio.security.service.AuthService;
import com.fatowlstudio.upload.service.UploadModelService;
import com.fatowlstudio.upload.service.UploadService;
import org.apache.log4j.Logger;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

/**
 * Default implementation for {@link UserDetailsService} interface. See the interface for public API documentation.
 */
@Service
@Transactional
public class UserDetailsServiceImpl implements UserDetailsService {

    private final Logger LOGGER = Logger.getLogger(this.getClass());

    private final UserDetailsRepository userDetailsRepository;
    private final AccountService accountService;
    private final UploadModelService uploadModelService;
    private final UploadService uploadService;
    private final AuthService authService;

    public UserDetailsServiceImpl(UserDetailsRepository userDetailsRepository, AccountService accountService,
        UploadModelService uploadModelService, UploadService uploadService, AuthService authService) {
        this.userDetailsRepository = userDetailsRepository;
        this.accountService = accountService;
        this.uploadModelService = uploadModelService;
```

```
        this.uploadService = uploadService;
        this.authService = authService;
    }

    @Override
    public UserDetailsDto findUser(final long userId) {
        return UserDetailsMapper.toDto(userDetailsRepository.findOne(userId));
    }

    @Override
    public UserDetailsDto createUser(final UserRegistrationDto userRegistrationForm) {
        // Check if user with provided phone already exists. If not, just register, if so, throw an exception
        UserDetails newUser = userDetailsRepository.findByPhone(userRegistrationForm.getPhone());
        if (newUser != null) {
            updateUserLocation(newUser.getId(), userRegistrationForm.getLocation());
            return updateUser(newUser.getId(), new UserUpdateDto(
                userRegistrationForm.getPhone(),
                userRegistrationForm.getName(),
                userRegistrationForm.getGender(),
                userRegistrationForm.getAvatar(),
                userRegistrationForm.isNeedy(),
                userRegistrationForm.isHelper(),
                newUser.getDescription(),
                newUser.getNotificationRadius(),
                newUser.isNotificationsActive()
            ));
        }

        newUser = new UserDetails();

        newUser.setLocation(userRegistrationForm.getLocation());
        newUser.setPhone(userRegistrationForm.getPhone());
        newUser.setName(userRegistrationForm.getName());
```

```
        newUser.setGender(userRegistrationForm.getGender());
        if (userRegistrationForm.getAvatar() != null) {
            newUser.setAvatar(userRegistrationForm.getAvatar());
            uploadModelService.confirm(userRegistrationForm.getAvatar());
        }
        newUser.setAuthorities(accountService.retrieveAuthorities(userRegistrationForm.isHelper(),
userRegistrationForm.isNeedy()));
        newUser.setDescription(null);
        newUser.setNotificationRadius(3000);
        newUser.setNotificationsActive(userRegistrationForm.isHelper());

        return UserDetailsMapper.toDto(userDetailsRepository.save(newUser));
    }

    @Override
    public UserDetailsDto updateUser(final long userId, final UserUpdateDto userUpdateForm) {
        UserDetails toUpdate = userDetailsRepository.findOne(userId);

        if (!accountService.isPhoneAvailableForUser(userId, userUpdateForm.getPhone()))
            throw new LoginNotAvailableException();

        toUpdate.setPhone(userUpdateForm.getPhone());
        toUpdate.setName(userUpdateForm.getName());
        toUpdate.setGender(userUpdateForm.getGender());

        if (toUpdate.getAvatar() != null &&
!toUpdate.getAvatar().equals(userUpdateForm.getAvatar()))

uploadService.deleteFileWithModel(uploadModelService.findByHttpResourceLink(toUpdate.getA
vatar()));
        toUpdate.setAvatar(userUpdateForm.getAvatar());
        if (toUpdate.getAvatar() != null)
            uploadModelService.confirm(userUpdateForm.getAvatar());

        toUpdate.setAuthorities(accountService.retrieveAuthorities(userUpdateForm.isHelper(),
userUpdateForm.isNeedy()));
```

```
        toUpdate.setDescription(userUpdateForm.getDescription());
        toUpdate.setNotificationRadius(userUpdateForm.getNotificationRadius());
        toUpdate.setNotificationsActive(userUpdateForm.isNotificationActive());

        return UserDetailsMapper.toDto(userDetailsRepository.save(toUpdate));
    }

    @Override
    public void updateUserLocation(final long userId, final Location location) {
        UserDetails toUpdate = userDetailsRepository.findOne(userId);
        toUpdate.setLocation(location);
        userDetailsRepository.save(toUpdate);
    }

    @Override
    public void updateFcmToken(String fcmToken) {
        userDetailsRepository.removeFcmToken(fcmToken);
        userDetailsRepository.updateFcmToken(authService.getLoggedUserPrincipal().getUserId(),
fcmToken);
    }
}
```

UserDetailsService

```
package com.fatowlstudio.core.service;

import com.fatowlstudio.core.model.Location;
import com.fatowlstudio.core.model.UserDetails;
import com.fatowlstudio.core.model.dto.UserDetailsDto;
import com.fatowlstudio.core.model.dto.UserRegistrationDto;
import com.fatowlstudio.core.model.dto.UserUpdateDto;

/**
 * Service responsible for operations on users.
```

```
 */
public interface UserDetailsService {

    /**
     * Retrieves single user with passed userId.
     *
     * @param userId id of the user to retrieve
     * @return user with passed id
     */
    UserDetailsDto findUser(long userId);

    /**
     * Creates in the database an user passed as a user parameter.
     *
     * @param userDetailsForm user to be persisted
     * @return representation of the persisted user
     */
    UserDetailsDto createUser(UserRegistrationDto userDetailsForm);

    /**
     * Updates in the database an user with passed userId.
     *
     * @param userId id of the user to update
     * @param userUpdateForm user updated data
     * @return representation of the updated user
     */
    UserDetailsDto updateUser(long userId, UserUpdateDto userUpdateForm);

    /**
     * Updates location of the user with passed userId.
     *
     * @param userId id of the user to update
     * @param location new location of the user
     */
```

```
    void updateUserLocation(long userId, Location location);

    /**
     * Updates logged user fcm token. If someone already has that token it will be first removed, so one device token
     * can be bind with only one account.
     *
     * @param fcmToken fcmToken to be set
     */
    void updateFcmToken(String fcmToken);
}
```

Task

OldHelpRequestPurger:

```
package com.fatowlstudio.core.task;

import com.fatowlstudio.core.service.HelpRequestService;
import org.springframework.scheduling.annotation.Scheduled;
import org.springframework.stereotype.Component;
import org.springframework.transaction.annotation.Transactional;

/**
 * CRON job responsible for removing old help requests. It will delete each old request older than two days.
 */
@Component
public class OldHelpRequestPurger {

    private final HelpRequestService helpRequestService;

    public OldHelpRequestPurger(HelpRequestService helpRequestService) {
        this.helpRequestService = helpRequestService;
    }
```

```
    /**
     * Purges old help requests every day at 2:00 a.m.
     */
    @Scheduled(cron = "0 0 2 * * *")
    @Transactional
    void purgeOldHelpRequests() {
        helpRequestService.findOldHelpRequests().forEach(helpRequestService::deleteHelpRequest);
    }
}
```

Utils

CodingUtils

```
package com.fatowlstudio.core.utils;

import org.apache.commons.lang3.StringUtils;

import java.nio.charset.StandardCharsets;
import java.util.Base64;
import java.util.UUID;

/**
 * Abstract helper class providing some commonly used logic for encoding and decoding Strings.
 */
public abstract class CodingUtils {

    /**
     * Decodes provided base64 string to human readable utf8 form.
     *
     * @param text base64 string to be decoded
     * @return decoded string in utf8
     */
```

```
    public static String decodeFromBase64(String text) {
        byte[] decodedText = Base64.getDecoder().decode(text);
        return new String(decodedText, StandardCharsets.UTF_8);
    }

    /**
     * Encodes provided utf8 string to base64 form of it.
     *
     * @param text utf8 string to be encoded
     * @return base64 encoded string
     */
    public static String encodeToBase64(String text) {
        return Base64.getEncoder().encodeToString(text.getBytes(StandardCharsets.UTF_8));
    }

    /**
     * Generates short UUID, which is first 8 letters of regular UUID.
     *
     * @return first 8 letters of generated UUID
     */
    public static String generateShortUUID() {
        return UUID.randomUUID().toString().split("-")[0];
    }

    /**
     * Generates long UUID, which is just regular java randomUUID. Method created for consistency sake.
     *
     * @return UUID
     */
    public static String generateLongUUID() {
        return UUID.randomUUID().toString();
    }
```

```
    /**
     * Removes diacritics (accents) from a string. The case will not be altered. It calls standard apache
     * {@link StringUtils#stripAccents(String)} method and then also replaces some characters that are not
     * properly handled by it.
     *
     * @param text text to be stripped
     * @return input text with diacritics removed
     */
    public static String stripAccents(String text) {
        return StringUtils.stripAccents(text)
                .replaceAll("Ł", "L")
                .replaceAll("ł", "l")
                .replaceAll("Ø", "O");
    }
}
```

PageRequestBuilder

```
package com.fatowlstudio.core.utils;

import org.springframework.data.domain.PageRequest;

import java.util.HashMap;
import java.util.Map;

/**
 * Builder for {@link org.springframework.data.domain.PageRequest} object used in Spring Data pagination.
 */
public abstract class PageRequestBuilder {

    public static final String PAGE_NUM_KEY = "pageNum";
    public static final String PER_PAGE_ELEMENTS_KEY = "perPageElements";
```

```
    /**
     * Builds {@link PageRequest} from passed page string and sort string. Page string should be in form of x,y where x is
     * number of requested page and y is elements per page, for example to get 3rd page with 10 elements pageString should
     * be 2,10 (pages are counted from 0). See {@link SortRequestBuilder#build(String)} for more information about sort
     * string. Sort string may be null, in that case no sorting will be built.
     *
     * @param pageString page string from which page request will be build
     * @param sortString sort string from which sorting options will be build
     * @return PageRequest object
     */
    public static PageRequest build(String pageString, String sortString) {
        Map<String, Integer> pageParams = getPageParams(pageString);
        int pageNum = pageParams.get(PAGE_NUM_KEY);
        int perPageElements = pageParams.get(PER_PAGE_ELEMENTS_KEY);
        if (sortString == null)
            return new PageRequest(pageNum, perPageElements);
        else
            return new PageRequest(pageNum, perPageElements, SortRequestBuilder.build(sortString));
    }


    /**
     * Creates a map with parameters for paging. Number of requested page will be stored at the {@link PageRequestBuilder#PAGE_NUM_KEY}
     * and requested elements per page will be stored at the {@link PageRequestBuilder#PER_PAGE_ELEMENTS_KEY}.
     *
     * @param pageString page string from which map parameters will be build
     * @return map with parameters for paging
     */
    private static Map<String, Integer> getPageParams(String pageString) {
        String[] pageStringParts = pageString.split(",");
```

```
        Map<String, Integer> pageParams = new HashMap<>();
        pageParams.put(PAGE_NUM_KEY, Integer.parseInt(pageStringParts[0]));
        pageParams.put(PER_PAGE_ELEMENTS_KEY, Integer.parseInt(pageStringParts[1]));
        return pageParams;
    }
}
```

SortRequestBuilder

```
package com.fatowlstudio.core.utils;

import org.springframework.data.domain.Sort;

/**
 * Builder for {@link org.springframework.data.domain.Sort} object used in Spring Data pagination.
 */
public abstract class SortRequestBuilder {

    /**
     * Builds {@link Sort} from passed sort string. Sort string should be in form of x:y where x is the field name and y
     * is the sorting order, eg. to sort descending by id sort string should be id:DESC. More than one sorting can be
     * specified, each sorting option should be separated by a comma, eg. id:DESC,author:ASC,rating:ASC.
     *
     * @param sortString sort string from which sorting options will be build
     * @return Sort object
     */
    public static Sort build(String sortString) {
        if (sortString == null || sortString.isEmpty())
            return new Sort(Sort.Direction.ASC, "id");

        // Get first sorting before loop as we can't create empty Sort to do it later with loop/stream
```

```
        String[] sorts = sortString.split(",");
        String[] firstSortParts = sorts[0].split(":");
        Sort sortObject = new Sort(directionFrom(firstSortParts[1]), firstSortParts[0]);

        for (int i = 1; i < sorts.length; i++) {
            String[] sortParts = sorts[i].split(":");
            sortObject = sortObject.and(new Sort(directionFrom(sortParts[1]), sortParts[0]));
        }

        return sortObject;
    }

    /**
     * Transforms string literals asc or desc into appropriate equivalent of the {@link
org.springframework.data.domain.Sort.Direction}
     * type.
     *
     * @param directionString literal asc or desc
     * @return Springs Direction object for provided directionString
     */
    private static Sort.Direction directionFrom(String directionString) {
        if (directionString.toLowerCase().equals("asc"))
            return Sort.Direction.ASC;
        if (directionString.toLowerCase().equals("desc"))
            return Sort.Direction.DESC;

        return null;
    }
}
```

Security

Api

AuthController

```
package com.fatowlstudio.security.api;

import com.fatowlstudio.core.api.BaseController;
import com.fatowlstudio.core.model.wrapper.Wrapper;
import com.fatowlstudio.security.model.LoginRequest;
import com.fatowlstudio.security.service.AuthService;
import io.swagger.annotations.ApiOperation;
import org.springframework.http.HttpStatus;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import javax.validation.Valid;

/**
 * Controller responsible for logging users in and out.
 */
@RestController
public class AuthController extends BaseController {

    private final AuthService authService;

    /**
     * Standard constructor.
     *
     * @param authService autowired authService
     */
    public AuthController(AuthService authService) {
        this.authService = authService;
```

```
    }

    /**
     * API bridge for {@link AuthService#authorize(LoginRequest)}. See it for more details.
     */
    @ApiOperation(
        value = "login",
        notes = "Generates JWT token for user with <b>login</b> and <b>password</b> provided as a LoginRequest and logs " +
                "that user in. Will return 401 in case of login failure."
    )
    @PostMapping(
        value = BaseController.LOGIN,
        consumes = MediaType.APPLICATION_JSON_VALUE,
        produces = MediaType.APPLICATION_JSON_VALUE
    )
    public ResponseEntity<Wrapper<String>> login(
        @Valid @RequestBody LoginRequest loginRequest
    ) {
        String token = authService.authorize(loginRequest);
        if (token != null)
            return new ResponseEntity<>(new Wrapper<>(token), HttpStatus.OK);
        else
            return new ResponseEntity<>(HttpStatus.UNAUTHORIZED);
    }

    /**
     * API bridge for {@link AuthService#logout()}. See it for more details.
     */
    @ApiOperation(
        value = "logout",
        notes = "Devalidates provided token, effectively logging owning it user off."
    )
    @PostMapping(
```

```
        value = BaseController.LOGOUT,
        consumes = MediaType.APPLICATION_JSON_VALUE
    )
    @PreAuthorize("@permissionEvaluator.anyAuthenticated()")
    public void logout() {
        authService.logout();
    }
}
```

Config

WebSecurityConfig

```
package com.fatowlstudio.security.config;

import com.fatowlstudio.core.api.BaseController;
import com.fatowlstudio.security.filter.JwtAuthenticationFilter;
import com.fatowlstudio.security.provider.JwtAuthenticationProvider;
import com.fatowlstudio.upload.model.value.Upload;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.http.HttpMethod;
import org.springframework.security.authentication.AuthenticationProvider;
import
org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuil
der;
import
org.springframework.security.config.annotation.method.configuration.EnableGlobalMethodSecurit
y;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.builders.WebSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import
org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;
import org.springframework.security.config.http.SessionCreationPolicy;
```

```
import org.springframework.security.web.AuthenticationEntryPoint;
import org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;
import org.springframework.web.servlet.config.annotation.CorsRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurerAdapter;

import javax.servlet.http.HttpServletResponse;

/**
 * Web security configuration class. This class defines global security rules, configures user authentication options
 * and integrates everything with Spring Security. It also contains some bean definitions needed for JWT in order to
 * work with Spring Security.
 */
@Configuration
@EnableWebSecurity
@EnableGlobalMethodSecurity(prePostEnabled = true)
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {

    // Name of the HTTP header containing JWT token
    public final static String JWT_TOKEN_HEADER = "X-Authorization";

    // Cost to determine number of BCrypt rounds for BCrypt password hashing
    public final static int BCRYPT_COST = 12;

    @Value("${app.swagger.enabled}")
    private boolean swaggerEnabled;

    /**
     * Security configuration method. It defines global rules for Spring Security. Secured endpoints are handled by
     * @PreAuthorize rule at controller method level so they should not be defined here. Endpoints with no @PreAuthorize
     * at method are validated against @PreAuthorize(isAuthenticated()) by default. If you want to make some method
```

```
    * accessible without authentication define it in the {@link WebSecurityConfig#configure(WebSecurity)} method as
    * ignored (see javadoc of it for more details).
    *
    * @param http the {@link HttpSecurity} to modify
    * @throws Exception if an error occurs
    */
   @Override
   protected void configure(HttpSecurity http) throws Exception {
      http
         .cors()
      .and()
         .csrf().disable()   // We don't need CSRF protection for JWT based authentication
         .exceptionHandling().authenticationEntryPoint(restAuthenticationEntryPoint())
      .and()
         .sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS)
      .and()
         .authorizeRequests()
            .anyRequest().fullyAuthenticated()
      .and()
         .addFilterBefore(new JwtAuthenticationFilter(authenticationManager()), UsernamePasswordAuthenticationFilter.class);
   }


   /**
    * Web security configuration method. It is used to declare paths that should not be verified by Spring Security.
    * It may be useful for enabling login/register endpoints for everyone or allowing resources to load without
    * authentication (js, html, css, image files etc).
    *
    * @param webSecurity the {@link WebSecurity} to modify
    */
   @Override
   public void configure(WebSecurity webSecurity) {
```

```
        // Ignore publicly available endpoints
        webSecurity.ignoring()
                .antMatchers(HttpMethod.POST, BaseController.LOGIN)
                .antMatchers(HttpMethod.POST, BaseController.USERS)
                .antMatchers(HttpMethod.POST, BaseController.UPLOADS)
                .antMatchers(Upload.Dirs.RESOURCE_HTTP_PATH);

        // Ignore swagger endpoints and resources - for development purposes
        if (swaggerEnabled)
            webSecurity.ignoring().antMatchers(
                    "/v2/api-docs",
                    "/configuration/ui",
                    "/swagger-resources/**",
                    "/configuration/security",
                    "/swagger-ui.html",
                    "/webjars/**"
            );
    }

    /**
     * Configures authentication manager. All possible authentication providers should be registered here.
     *
     * @param authBuilder Spring Security AuthenticationManagerBuilder to configure
     */
    @Override
    protected void configure(AuthenticationManagerBuilder authBuilder) {
        authBuilder.authenticationProvider(defaultAuthenticationProvider());
    }

    /**
     * Entry point for users without token. We declare it to override default redirection to the login page as
     * for REST services default behavior doesn't make sense. Instead of redirect it just returns 401 code.
```

```
     *
     * @return 401 UNAUTHORIZED HTTP status
     */
    @Bean
    public AuthenticationEntryPoint restAuthenticationEntryPoint() {
        return (request, response, authException) -> response.sendError(HttpServletResponse.SC_UNAUTHORIZED);
    }

    /**
     * Returns default authentication provider.
     *
     * @return default authentication provider
     */
    @Bean
    public AuthenticationProvider defaultAuthenticationProvider() {
        return new JwtAuthenticationProvider();
    }

    /**
     * CORS filter bean.
     *
     * @return CORS filter bean
     */
    @Bean
    public WebMvcConfigurer corsConfigurer() {
        return new WebMvcConfigurerAdapter() {
            @Override
            public void addCorsMappings(CorsRegistry registry) {
                registry
                    .addMapping("/**")
                    .allowCredentials(false)
                    .allowedHeaders("Content-Type", "Origin", "Accept", "X-Authorization")
                    .allowedMethods("GET", "POST", "DELETE", "PUT", "OPTIONS")
```

```
                .allowedOrigins("*");
        }
    };
  }
}
```

Ex

InvalidJwtSignatureException

```
package com.fatowlstudio.security.ex;

import org.springframework.security.core.AuthenticationException;

/**
 * Exception signalising that signature of parsed JWT token is not valid. To be thrown in Spring Security
 * authentication providers.
 */
public class InvalidJwtSignatureException extends AuthenticationException {

  public InvalidJwtSignatureException(String msg, Throwable t) {
    super(msg, t);
  }
}
```

JwtProcessingException

```
package com.fatowlstudio.security.ex;

import org.springframework.security.core.AuthenticationException;

/**
 * Exception signalising general issue with JWT token processing. To be thrown in Spring Security
```

```
 * authentication providers.
 */
public class JwtProcessingException extends AuthenticationException {

    public JwtProcessingException(String msg, Throwable t) {
        super(msg, t);
    }
}
```

Filter

JwtAuthenticationFilter:

```
package com.fatowlstudio.security.filter;

import com.fatowlstudio.security.config.WebSecurityConfig;
import com.fatowlstudio.security.model.JwtTokenAuthorizationRequest;
import org.apache.log4j.Logger;
import org.springframework.http.HttpStatus;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.AuthenticationException;
import org.springframework.security.core.context.SecurityContext;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.web.authentication.AbstractAuthenticationProcessingFilter;

import javax.servlet.FilterChain;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

/**
```

```
 * Authentication filter responsible for extracting JWT token from request and passing it to authentication manager for
 * validation. It will fill Spring Security context with authenticated Authentication object in case of successful
 * authentication or clear security context otherwise.
 */
public class JwtAuthenticationFilter extends AbstractAuthenticationProcessingFilter {

    private final Logger LOGGER = Logger.getLogger(this.getClass());

    /**
     * Standard constructor. Paths that should be protected by this filter are set in call to the superclass constructor.
     */
    public JwtAuthenticationFilter(AuthenticationManager authenticationManager) {
        super("/**");
        setAuthenticationManager(authenticationManager);
    }

    /**
     * Attempts to authenticate user. It will try to authenticate all requests to all endpoints besides those declared
     * as 'permit all' in the {@link WebSecurityConfig}. Authentication is based on JWT token passed in the X-Authorization
     * HTTP header and with {@link com.fatowlstudio.security.provider.JwtAuthenticationProvider} provider. Based on
     * authentication result it will pass security chain execution to appropriate success/failure handler.
     *
     * @param request http request
     * @param response http response
     * @return authenticated Spring Security authentication object in case of successful authentication
     * @throws AuthenticationException in case of authentication failure
     * @throws IOException in case of io error
     * @throws ServletException in case of servlet error
     */
```

```
    @Override
    public Authentication attemptAuthentication(HttpServletRequest request, HttpServletResponse response)
            throws AuthenticationException, IOException, ServletException {
        String rawToken = request.getHeader(WebSecurityConfig.JWT_TOKEN_HEADER);
        return getAuthenticationManager().authenticate(new JwtTokenAuthorizationRequest(rawToken));
    }

    /**
     * Processes successfully authenticated request. Basically it will fill security context with authentication result
     * retrieved from {@link JwtAuthenticationFilter#attemptAuthentication} method and pass request for further filtering.
     *
     * @param request http request
     * @param response http response
     * @param chain security chain
     * @param authResult result of the successful authentication
     * @throws IOException in case of IO error
     * @throws ServletException  in case of servlet error
     */
    @Override
    protected void successfulAuthentication(HttpServletRequest request, HttpServletResponse response, FilterChain chain,
            Authentication authResult) throws IOException, ServletException {
        SecurityContext securityContext = SecurityContextHolder.createEmptyContext();
        securityContext.setAuthentication(authResult);
        SecurityContextHolder.setContext(securityContext);
        chain.doFilter(request, response);
    }

    /**
     * Process unsuccessfully authenticated request. Basically it will clear security context and return 401 UNAUTHORIZED
     * HTTP status.
```

```
     *
     * @param request http request
     * @param response http response
     * @param authException authorization ex
     * @throws IOException in case of IO error
     * @throws ServletException in case of servlet error
     */
    @Override
    protected void unsuccessfulAuthentication(HttpServletRequest request, HttpServletResponse response, AuthenticationException authException)
            throws IOException, ServletException {
        LOGGER.trace("Request failed authentication: " + request.getServletPath());
        SecurityContextHolder.clearContext();
        response.setStatus(HttpStatus.UNAUTHORIZED.value());
    }
}
```

Model

Authority

```
package com.fatowlstudio.security.model;

import com.fatowlstudio.security.service.PermissionEvaluator;
import org.springframework.stereotype.Component;

/**
 * Container for storing default authorities values. This class has to be non-abstract so Spring can put it into container
 * for later use by {@link PermissionEvaluator} but it's constructor is private so it can't be declared explicitly in
 * code by parties other than Spring.
 */
@Component("authority")
public class Authority {
```

```
    private Authority() {

    }

    public static final String ADMIN = "ADMIN";
    public static final String USER = "USER";
    public static final String NEEDY = "NEEDY";
    public static final String HELPER = "HELPER";
}
```

JwtToken

```
package com.fatowlstudio.security.model;

import java.util.Date;

/**
 * Model for storing object representation of the JWT token. Its immutable.
 */
public class JwtToken {

    private String token;

    private String subject;

    private String authorities;

    private String tokenId;

    private Date issuedAt;

    private long userId;
```

```
    /**
     * Standard constructor
     *
     * @param token raw JWT token
     * @param subject sub claim
     * @param authorities rol claim
     * @param tokenId jti claim
     * @param issuedAt iat claim
     * @param userId uid claim
     */
    public JwtToken(String token, String subject, String authorities, String tokenId, Date issuedAt, long userId) {
        this.token = token;
        this.subject = subject;
        this.authorities = authorities;
        this.tokenId = tokenId;
        this.issuedAt = issuedAt;
        this.userId = userId;
    }

    public String getToken() {
        return token;
    }

    public String getSubject() {
        return subject;
    }

    public String getAuthorities() {
        return authorities;
    }

    public String getTokenId() {
        return tokenId;
```

```
    }

    public Date getIssuedAt() {
        return issuedAt;
    }

    public long getUserId() {
        return userId;
    }
}
```

JwtTokenAuthorizationRequest

```
package com.fatowlstudio.security.model;

import org.springframework.security.authentication.AbstractAuthenticationToken;
import org.springframework.security.core.GrantedAuthority;

import java.util.Collection;

/**
 * Spring Security authentication class. It holds authentication data used when authenticating by
 * {@link com.fatowlstudio.security.provider.JwtAuthenticationProvider} and by Spring Security chain after
 * successful authorization.
 */
public class JwtTokenAuthorizationRequest extends AbstractAuthenticationToken {

    private String rawJwtToken;
    private JwtToken objectJwtToken;

    /**
     * Constructor used when passing object to be authenticated. Its called in
     * {@link com.fatowlstudio.security.filter.JwtAuthenticationFilter#attemptAuthentication} and works with
```

```
    * {@link com.fatowlstudio.security.provider.JwtAuthenticationProvider} where token is parsed and validated.
    * JwtAuthenticationProvider will return new instance of it with authorities set by another constructor.
    * See appropriate authentication provider for more details.
    *
    * @param rawJwtToken string representing raw jwt token retrieved form request
    */
   public JwtTokenAuthorizationRequest(String rawJwtToken) {
      super(null);
      this.rawJwtToken = rawJwtToken;
      this.objectJwtToken = null;
      this.setAuthenticated(false);
   }

   /**
    * Constructor used by {@link com.fatowlstudio.security.provider.JwtAuthenticationProvider} to fill authorities
    * and mark this object as authenticated after successful token processing in the provider.
    *
    * @param jwtToken object representation of parsed raw jwt token
    * @param authorities authorities of the successfuly validated token
    */
   public JwtTokenAuthorizationRequest(JwtToken jwtToken, Collection<? extends GrantedAuthority> authorities) {
      super(authorities);
      this.eraseCredentials();
      this.objectJwtToken = jwtToken;
      super.setAuthenticated(true);
   }

   @Override
   public void setAuthenticated(boolean authenticated) {
      if (authenticated)
         throw new IllegalArgumentException("Token cannot be set to trusted state outside of the appropriate authentication provider");
```

```
        super.setAuthenticated(false);
    }

    @Override
    public Object getCredentials() {
        return rawJwtToken;
    }

    @Override
    public Object getPrincipal() {
        return objectJwtToken;
    }

    @Override
    public void eraseCredentials() {
        super.eraseCredentials();
        rawJwtToken = null;
    }
}
```

LoginRequest

```
package com.fatowlstudio.security.model;

/**
 * Model representing login request. Its immutable. One of the two: password or facebookAccessToken is mandatory. If
 * both provided the facebookAccessToken will take precedence and login request will be treated as social login request.
 */
public class LoginRequest {

    private String login;

    public LoginRequest() {
```

```
    }

    public LoginRequest(String login, String password) {
        this.login = login;
    }

    /*
     * GETTERS AND SETTERS
     */

    public String getLogin() {
        return login;
    }
}
```

TokenStorage

```
package com.fatowlstudio.security.model;

import java.util.Collections;
import java.util.HashMap;
import java.util.Map;

/**
 * Abstract class being in-memory container for tokens signed by the server. Every created token has its id, which is
 * stored here. When request arrives JWT provided with it is checked against base of signed tokens - if its not found it
 * will be treated as invalid. Logging out also removes token of logged out user from this storage disabling further use
 * of that token. This class contains some static helper methods to operate on storage.
 */
public abstract class TokenStorage {
```

```
    /**
     * Token storage. Its a map of tokens, where the key is a token (tokens are unique) and value is an
id of the user
     * to which this token belongs.
     */
    private static Map<String, Long> tokenStorage = new HashMap<>();

    /**
     * Saves token in the token store.
     *
     * @param token token to be saved
     * @param userId id of the user to which this token belongs
     */
    public static void putTokenInStore(String token, Long userId) {
        tokenStorage.put(token, userId);
    }

    /**
     * Removes token from the token store. If there is no such token nothing will happen.
     *
     * @param token token id to be removed
     */
    public static void removeTokenFromStore(String token) {
        if (contains(token))
            tokenStorage.remove(token);
    }

    /**
     * Removes from the token store all tokens belonging to the user with passed user id.
     *
     * @param userId id of the user whose tokens should be removed
     */
    public static void removeAllUserTokensFromStore(Long userId) {
        tokenStorage.values().removeAll(Collections.singleton(userId));
```

```
    }

    /**
     * Checks if token storage contains provided token.
     *
     * @param token token id to be checked
     * @return true if storage contains that id, else otherwise
     */
    public static boolean contains(String token) {
        return tokenStorage.containsKey(token);
    }
}
```

Provider

JwtAuthenticationProvider

```
package com.fatowlstudio.security.provider;

import com.fatowlstudio.security.ex.InvalidJwtSignatureException;
import com.fatowlstudio.security.ex.JwtProcessingException;
import com.fatowlstudio.security.model.JwtToken;
import com.fatowlstudio.security.model.JwtTokenAuthorizationRequest;
import com.fatowlstudio.security.model.TokenStorage;
import com.fatowlstudio.security.service.TokenService;
import io.jsonwebtoken.JwtException;
import io.jsonwebtoken.SignatureException;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.authentication.AuthenticationCredentialsNotFoundException;
import org.springframework.security.authentication.AuthenticationProvider;
import org.springframework.security.authentication.InsufficientAuthenticationException;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.AuthenticationException;
import org.springframework.security.core.authority.AuthorityUtils;
```

```
/**
 * Spring Security authentication provider for {@link com.fatowlstudio.security.model.JwtTokenAuthorizationRequest} class.
 * Its responsible for creating authenticated Authentication object from provided JwtTokenAuthorizationRequest and pass it
 * for further processing in Spring Security chain. It have to be registered and called from {@link com.fatowlstudio.security.filter.JwtAuthenticationFilter}
 *
 * Fields of this class are autowired at field level because Spring Security filters are instantiated before regular beans,
 * so they wont be available at the time of this filter constructor call for constructor injection. They will be injected
 * in a later lifecycle phase when they are ready.
 */
@SuppressWarnings({"SpringAutowiredFieldsWarningInspection", "SpringJavaAutowiredMembersInspection"})
public class JwtAuthenticationProvider implements AuthenticationProvider {

    @Autowired
    private TokenService tokenService;

    /**
     * Validates provided authentication object.
     *
     * @param authentication authentication object to be validated
     * @return new authentication object filled with authorities in case of successful validation
     * @throws AuthenticationException in case of validation error
     */
    @Override
    public Authentication authenticate(Authentication authentication) throws AuthenticationException {
        String rawToken = (String) authentication.getCredentials();
        if (rawToken == null || rawToken.equals("") || !rawToken.startsWith(tokenService.getTokenPrefix()))
            throw new InsufficientAuthenticationException("No authentication data were found");
```

```
        JwtToken jwtToken;
        try {
            jwtToken = tokenService.parseToken(rawToken);
            if (!TokenStorage.contains(jwtToken.getToken()))
                throw new AuthenticationCredentialsNotFoundException("The signature of provided token is valid, but it is " +
                        "not stored in signed tokens storage - probably user logged off or token expired. Try logging in again.");
        } catch (SignatureException e) {
            throw new InvalidJwtSignatureException("JWT signature verification failed.", e);
        } catch (JwtException e) {
            throw new JwtProcessingException("Authentication ex while parsing token - make sure provided token is " +
                    "properly constructed and filled with all required data.", e);
        }

        return new JwtTokenAuthorizationRequest(jwtToken,
                AuthorityUtils.commaSeparatedStringToAuthorityList(jwtToken.getAuthorities()));
    }

    /**
     * Checks if this authentication provider is a valid provider for an authentication object passed to it by Spring Security.
     *
     * @param authenticationClass class of authentication object
     * @return true if this provider can authenticate object of provided class
     */
    @Override
    public boolean supports(Class<?> authenticationClass) {
        return authenticationClass.equals(JwtTokenAuthorizationRequest.class);
    }
}
```

Service

AuthService:

```
package com.fatowlstudio.security.service;

import com.fatowlstudio.core.model.UserDetails;
import com.fatowlstudio.security.model.JwtToken;
import com.fatowlstudio.security.model.LoginRequest;

/**
 * Service responsible for authorization, logging users in nad out.
 */
public interface AuthService {

    /**
     * Generates JWT token for user with username and password provided as a {@link LoginRequest} and logs that user in.
     *
     * @param loginRequest object representing user that want to log in
     * @return JWT token if user successfully logged in, null otherwise
     */
    String authorize(LoginRequest loginRequest);

    /**
     * Devalidates token of user making this request, effectively logging owning it user off.
     */
    void logout();

    /**
     * Retrieves principal of user making request. Principal is {@link JwtToken} class. May return null if requesting user
     * is anonymous.
     *
     * @return requesting user security principal or null if requesting user is anonymous
     */
    JwtToken getLoggedUserPrincipal();
```

```
    /**
     * Retrieves {@link UserDetails} of user making request. May return null if requesting user is anonymous.
     *
     * @return requesting user details or null if requesting user is anonymous
     */
    UserDetails getLoggedUser();
}
```

Impl

AuthServiceImpl:

```
package com.fatowlstudio.security.service.impl;

import com.fatowlstudio.core.model.UserDetails;
import com.fatowlstudio.core.repository.UserDetailsRepository;
import com.fatowlstudio.security.model.JwtToken;
import com.fatowlstudio.security.model.LoginRequest;
import com.fatowlstudio.security.model.TokenStorage;
import com.fatowlstudio.security.service.AuthService;
import com.fatowlstudio.security.service.TokenService;
import org.apache.log4j.Logger;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.crypto.bcrypt.BCrypt;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

/**
 * Default implementation for {@link com.fatowlstudio.security.service.AuthService} interface. See the interface for
 * public API documentation.
 */
```

```
@Service
@Transactional
public class AuthServiceImpl implements AuthService {

    private final Logger LOGGER = Logger.getLogger(this.getClass());

    private final UserDetailsRepository userDetailsRepository;

    private final TokenService tokenService;

    /**
     * Standard constructor.
     *
     * @param tokenService autowired tokenService
     * @param userDetailsRepository autowired userDetailsRepository
     */
    public AuthServiceImpl(TokenService tokenService, UserDetailsRepository
userDetailsRepository) {
        this.tokenService = tokenService;
        this.userDetailsRepository = userDetailsRepository;
    }

    @Override
    public String authorize(LoginRequest loginRequest) {
        return authorizeLoginPassword(loginRequest);
    }

    @Override
    public void logout() {
        TokenStorage.removeTokenFromStore(getLoggedUserPrincipal().getToken());
    }

    @Override
    public JwtToken getLoggedUserPrincipal() {
```

```
        Authentication authentication = SecurityContextHolder.getContext().getAuthentication();
        return authentication != null ? (JwtToken) authentication.getPrincipal() : null;
    }

    @Override
    public UserDetails getLoggedUser() {
        JwtToken loggedUserJwtToken = getLoggedUserPrincipal();
        return loggedUserJwtToken != null ? userDetailsRepository.findOne(loggedUserJwtToken.getUserId()) : null;
    }

    /**
     * Authorizes user based on its login and password.
     *
     * @param loginRequest login request containing login and password
     * @return token if user successfully authorized
     */
    private String authorizeLoginPassword(LoginRequest loginRequest) {
        UserDetails userDetails = userDetailsRepository.findByPhone(loginRequest.getLogin());

        if (userDetails == null)
            return null;

        return generateTokenAndLogIn(userDetails);
    }

    /**
     * Generates token for user with provided user details and logs that user in.
     *
     * @param userDetails details of user for which token should be generated
     * @return jwt token for passed user
     */
    private String generateTokenAndLogIn(UserDetails userDetails) {
        String generatedToken = tokenService.generateToken(userDetails);
```

```
        TokenStorage.putTokenInStore(generatedToken, userDetails.getId());
        return generatedToken;
    }
}
```

PermissionEvaluatorImpl:

```
package com.fatowlstudio.security.service.impl;

import com.fatowlstudio.core.repository.HelpRequestRepository;
import com.fatowlstudio.security.model.JwtToken;
import com.fatowlstudio.security.service.AuthService;
import com.fatowlstudio.security.service.PermissionEvaluator;
import org.apache.log4j.Logger;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import java.util.Arrays;
import java.util.List;

/**
 * Default implementation of the {@link PermissionEvaluator}. See it for public API documentation.
 */
@Service(value = "permissionEvaluator")
@Transactional
@Scope("prototype")
public class PermissionEvaluatorImpl implements PermissionEvaluator {

    private final Logger LOGGER = Logger.getLogger(this.getClass());

    private final boolean authenticatedUser;
    private final List<String> authorities;
```

```
    private final long requestingUserId;

    private final HelpRequestRepository helpRequestRepository;

    public PermissionEvaluatorImpl(AuthService authService, HelpRequestRepository helpRequestRepository) {
        JwtToken loggedUser = authService.getLoggedUserPrincipal();
        this.authenticatedUser = loggedUser != null;
        this.authorities = loggedUser != null ? getAuthoritiesAsList(authService.getLoggedUserPrincipal().getAuthorities()) : null;
        this.requestingUserId = loggedUser != null ? loggedUser.getUserId() : -1L;
        LOGGER.trace("Created new permission evaluator");

        this.helpRequestRepository = helpRequestRepository;
    }

    @Override
    public boolean anyOfRoles(String... roles) {
        return authenticatedUser && Arrays.stream(roles).anyMatch(authorities::contains);
    }

    @Override
    public boolean self(long targetId) {
        return authenticatedUser && requestingUserId == targetId;
    }

    @Override
    public boolean selfAndAnyOfRoles(long targetId, String... roles) {
        return self(targetId) && anyOfRoles(roles);
    }

    @Override
    public boolean selfOrAnyOfRoles(long targetId, String... roles) {
        return self(targetId) || anyOfRoles(roles);
    }
```

```
    @Override
    public boolean anyAuthenticated() {
        return authorities != null && authorities.size() > 0;
    }

    @Override
    public boolean requestOwner(long requestId) {
        return helpRequestRepository.finOwnerIdOfHelpRequest(requestId) == requestingUserId;
    }

    /**
     * Converts authorities comma separated string into list of those authorities. Eg. "USER, ADMIN" string will be converted
     * to list ["USER", "ADMIN"].
     *
     * @param authorities authorities string to be converted to list of authorities
     * @return list of authorities
     */
    private List<String> getAuthoritiesAsList(String authorities) {
        return Arrays.asList(authorities.replace(" ", "").split(","));
    }
}
```

TokenServiceImpl:

```
package com.fatowlstudio.security.service.impl;

import com.fatowlstudio.core.model.UserDetails;
import com.fatowlstudio.security.model.JwtToken;
import com.fatowlstudio.security.service.TokenService;
import com.fatowlstudio.core.utils.CodingUtils;
import io.jsonwebtoken.Claims;
import io.jsonwebtoken.Jwts;
```

```
import io.jsonwebtoken.SignatureAlgorithm;
import io.jsonwebtoken.SignatureException;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Service;

import java.time.Instant;
import java.util.Date;
import java.util.UUID;

/**
 * Default implementation for {@link com.fatowlstudio.security.service.TokenService} interface. See the interface for
 * public API documentation.
 */
@Service
public class TokenServiceImpl implements TokenService {

    /**
     * Base64 encoded secret key used for JWT tokens signing and verifying.
     */
    private final String TOKEN_KEY;

    /**
     * Prefix for the JWT tokens
     */
    private final String TOKEN_PREFIX = "Bearer ";

    /**
     * Standard constructor.
     */
    public TokenServiceImpl(@Value("${app.security.secret}") String tokenSecret) {
        this.TOKEN_KEY = CodingUtils.encodeToBase64(tokenSecret);
    }
```

```
@Override
public String generateToken(UserDetails userDetails) {
    return TOKEN_PREFIX + Jwts.builder()
        .setSubject(userDetails.getPhone())
        .setIssuedAt(Date.from(Instant.now()))
        .setId(UUID.randomUUID().toString())
        .claim("rol", userDetails.getAuthorities())
        .claim("uid", userDetails.getId())
        .signWith(SignatureAlgorithm.HS256, TOKEN_KEY)
        .compact();
}

@Override
public JwtToken parseToken(String rawToken) throws SignatureException {
    Claims claims = getClaims(withoutBearer(rawToken));
    return new JwtToken(
        rawToken,
        claims.getSubject(),
        claims.get("rol").toString(),
        claims.getId(),
        claims.getIssuedAt(),
        ((Integer) claims.get("uid")).longValue()
    );
}

@Override
public String getTokenId(String rawToken) {
    return getClaims(withoutBearer(rawToken)).getId();
}

@Override
public String getTokenPrefix() {
    return TOKEN_PREFIX;
}
```

```
    /**
     * Retrieves body (claims) section of the passed jwt token.
     *
     * @param rawToken JWT token in string form from which claims should be retrieved
     * @return Claims of the passed token
     */
    private Claims getClaims(String rawToken) {
        return Jwts.parser().setSigningKey(TOKEN_KEY).parseClaimsJws(rawToken).getBody();
    }

    /**
     * Strips any used token prefix from provided JWT token.
     *
     * @param rawToken token with prefix
     * @return token without prefix
     */
    private String withoutBearer(String rawToken) {
        return rawToken.substring(TOKEN_PREFIX.length());
    }
}
```

PermissionEvaluator

```
package com.fatowlstudio.security.service;

import com.fatowlstudio.core.model.HelpRequest;

/**
 * Interface for advanced code based permission evaluation. It is used instead of complicated spEL expressions or when
 * database query is needed due to evaluate expression.
 */
public interface PermissionEvaluator {
```

```
    /**
     * Permits requesting user to take action only if he has at least one of the roles specified in roles parameter.
     *
     * @param roles roles against which user should be checked
     * @return true if user has rights to take requested action, false otherwise
     */
    boolean anyOfRoles(String... roles);

    /**
     * Permits requesting user to take action only if requesting user id is equal to the passed target id. It's used
     * mostly as the helper method for {@link this#selfAndAnyOfRoles} and {@link this#selfOrAnyOfRoles(long, String...)}
     * but may be used independently.
     *
     * @param targetId id of the target of this request
     * @return true if user has rights to take requested action, false otherwise
     */
    boolean self(long targetId);

    /**
     * Permits requesting user to take action only if he has at least one of the roles specified in roles parameter AND
     * requesting user id is equal to the passed target id. It is used for example when user wants to update some data,
     * but he/she should be able to use only his/her own data.
     *
     * @param targetId id of the target of this request
     * @param roles roles against which user should be checked
     * @return true if user has rights to take requested action, false otherwise
     */
    boolean selfAndAnyOfRoles(long targetId, String... roles);

    /**
```

```
    * Permits requesting user to take action only if he has at least one of the roles specified in roles parameter OR
    * requesting user id is equal to the passed target id. It is used for example when user wants to delete something,
    * but as a regular user he/she should be able to delete only own data, while admin should be able to delete any data.
    *
    * @param targetId id of the target of this request
    * @param roles roles against which user should be checked
    * @return true if user has rights to take requested action, false otherwise
    */
   boolean selfOrAnyOfRoles(long targetId, String... roles);

   /**
    * Permits requesting user to take action if he is authenticated (has any role).
    *
    * @return true if user has rights to take requested action, false otherwise
    */
   boolean anyAuthenticated();

   /**
    * Permits requesting user to take action if he is an logged owner of {@link HelpRequest} with passed requestId.
    *
    * @param requestId id of the help request of which logged user should be owner
    * @return true if user has rights to take requested action, false otherwise
    */
   boolean requestOwner(long requestId);
}
```

TokenService

```
package com.fatowlstudio.security.service;

import com.fatowlstudio.core.model.UserDetails;
```

```
import com.fatowlstudio.security.model.JwtToken;

/**
 * Service responsible for operations on jwt tokens.
 */
public interface TokenService {

    /**
     * Generates JWT token for user with provided details.
     *
     * @param userDetails details of the user for which token should be created
     * @return JWT token for provided user
     */
    String generateToken(UserDetails userDetails);

    /**
     * Parses provided JWT token into object representing that JWT token or
     * throw ex in case of token being malformed.
     *
     * @param rawToken JWT token in its raw string form
     * @return JWT token object representing passed rawToken
     */
    JwtToken parseToken(String rawToken);

    /**
     * Retrieves token id (jit claim) from the raw token.
     *
     * @param rawToken raw token from which id should be retrieved
     * @return jit claim of the passed token
     */
    String getTokenId(String rawToken);

    /**
     * Returns prefix used for JWT tokens
```

```
     *
     * @return prefix used for JWT tokens
     */
    String getTokenPrefix();
}
```

Upload

Api

UploadController

```
package com.fatowlstudio.upload.api;

import com.fatowlstudio.core.api.BaseController;
import com.fatowlstudio.upload.model.UploadModel;
import com.fatowlstudio.upload.service.UploadService;
import io.swagger.annotations.ApiOperation;
import org.springframework.http.MediaType;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.multipart.MultipartFile;

import java.io.IOException;

/**
 * Controller responsible for uploading and managing files on the server.
 */
@RestController
public class UploadController extends BaseController {

    private final UploadService uploadService;
```

```
    /**
     * Standard constructor.
     *
     * @param uploadService autowired uploadService
     */
    public UploadController(UploadService uploadService) {
        this.uploadService = uploadService;
    }

    /**
     * API bridge for {@link UploadService#uploadFile(MultipartFile)}. See it for more info.
     */
    @ApiOperation(
        value = "upload",
        notes = "Uploads multipart file to the server. Internal file location in the host file system will be decided " +
                "based on uploaded file type. Full model of uploaded file will be returned. Allowed extensions are: jpg, " +
                "png, pdf, mp4. File type is discovered automatically."
    )
    @PostMapping(
        value = BaseController.UPLOADS,
        consumes = MediaType.MULTIPART_FORM_DATA_VALUE,
        produces = MediaType.APPLICATION_JSON_VALUE
    )
    public UploadModel upload(
        @RequestParam(value = "file", required = true) MultipartFile file
    ) throws IOException {
        return uploadService.uploadFile(file);
    }

}
```

Config

PortListener

```
package com.fatowlstudio.upload.config;

import org.apache.log4j.Logger;
import org.springframework.boot.context.embedded.EmbeddedServletContainerInitializedEvent;
import org.springframework.context.ApplicationListener;
import org.springframework.stereotype.Component;

/**
 * Listener class waiting for embedded servlet container to initialize so application port will be set. Port discovery is
 * done with help of this class as per time when @Value properties are injected port is not set and always equals 0. Set
 * port will be stored in the runningPort variable.
 */
@Component
public class PortListener implements ApplicationListener<EmbeddedServletContainerInitializedEvent> {

    private final Logger LOGGER = Logger.getLogger(this.getClass());

    public static int runningPort;

    @Override
    public void onApplicationEvent(EmbeddedServletContainerInitializedEvent event) {
        runningPort = event.getEmbeddedServletContainer().getPort();
        LOGGER.info("Embedded servlet container initialized, port set.");
    }
}
```

WebMvcConfig

```
package com.fatowlstudio.upload.config;
```

```
import com.fatowlstudio.upload.model.value.Upload;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurerAdapter;
import org.springframework.web.servlet.resource.PathResourceResolver;

/**
 * Web MVC configuration class. This class is used to configure resources paths and enable resources serving from Tomcat.
 */
@Configuration
public class WebMvcConfig extends WebMvcConfigurerAdapter {

    public void addResourceHandlers(ResourceHandlerRegistry registry) {
        registry.addResourceHandler(Upload.Dirs.RESOURCE_HTTP_PATH)
                .addResourceLocations("file:" + Upload.Dirs.RESOURCES_BASE + "/")
                .setCachePeriod(0)
                .resourceChain(false)
                .addResolver(new PathResourceResolver());
    }
}
```

Ex

InvalidDataException

```
package com.fatowlstudio.upload.ex;

import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.ResponseStatus;

/**
 * Exception thrown when user provided invalid data (that can't be validated with JSR validation and is catched
 * with application logic, like change password fields don't matching).
```

```
 */
@ResponseStatus(value = HttpStatus.BAD_REQUEST)
public class InvalidDataException extends RuntimeException {

    public InvalidDataException(String message) {
        super(message);
    }
}
```

Model

UploadModel

```
package com.fatowlstudio.upload.model;

import com.fasterxml.jackson.annotation.JsonIgnore;
import com.fatowlstudio.upload.model.value.Upload;

import javax.persistence.*;
import java.time.ZoneOffset;
import java.time.ZonedDateTime;

/**
 * Model for storing nad transferring data produced during file upload.
 */
@Entity
public class UploadModel {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private long id;

    @Column(columnDefinition = "text", nullable = false, unique = true)
    private String resourceHttpLink;
```

```
    @Column(columnDefinition = "text", nullable = false, unique = true)
    @JsonIgnore
    private String resourcePhysicalLocation;

    @Column(nullable = false)
    @Enumerated(EnumType.STRING)
    private Upload.Type uploadType;

    @Column(nullable = false)
    private String format;

    @Column(nullable = false)
    @JsonIgnore
    private boolean confirmed;

    @Column(nullable = false)
    @JsonIgnore
    private ZonedDateTime uploadDate;

    @Column(name = "kb_file_size")
    private long kBfileSize;

    private String fileName;

    public UploadModel() {
    }

    public UploadModel(String resourceHttpLink, String resourcePhysicalLocation, Upload.Type
uploadType, String format,
                       boolean confirmed, ZonedDateTime uploadDate, long kBfileSize, String fileName) {
        this.resourceHttpLink = resourceHttpLink;
        this.resourcePhysicalLocation = resourcePhysicalLocation;
        this.uploadType = uploadType;
```

```
        this.format = format;
        this.confirmed = confirmed;
        this.uploadDate = uploadDate;
        this.kBfileSize = kBfileSize;
        this.fileName = fileName;
    }

    public UploadModel(String resourceHttpLink, String resourcePhysicalLocation, Upload.Type
uploadType, String format,
                       long kBfileSize, String fileName) {
        this(resourceHttpLink, resourcePhysicalLocation, uploadType, format, false,
ZonedDateTime.now(ZoneOffset.UTC),
                kBfileSize, fileName);
    }

    public long getId() {
        return id;
    }

    public void setId(long id) {
        this.id = id;
    }

    public String getResourceHttpLink() {
        return resourceHttpLink;
    }

    public void setResourceHttpLink(String resourceHttpLink) {
        this.resourceHttpLink = resourceHttpLink;
    }

    public String getResourcePhysicalLocation() {
        return resourcePhysicalLocation;
    }
```

```
public void setResourcePhysicalLocation(String resourcePhysicalLocation) {
    this.resourcePhysicalLocation = resourcePhysicalLocation;
}

public Upload.Type getUploadType() {
    return uploadType;
}

public void setUploadType(Upload.Type uploadType) {
    this.uploadType = uploadType;
}

public String getFormat() {
    return format;
}

public void setFormat(String format) {
    this.format = format;
}

public boolean isConfirmed() {
    return confirmed;
}

public void setConfirmed(boolean confirmed) {
    this.confirmed = confirmed;
}

public ZonedDateTime getUploadDate() {
    return uploadDate;
}

public void setUploadDate(ZonedDateTime uploadDate) {
    this.uploadDate = uploadDate;
```

```
    }

    public long getkBfileSize() {
        return kBfileSize;
    }

    public void setkBfileSize(long kBfileSize) {
        this.kBfileSize = kBfileSize;
    }

    public String getFileName() {
        return fileName;
    }

    public void setFileName(String fileName) {
        this.fileName = fileName;
    }
}
```

Value

Upload

```
package com.fatowlstudio.upload.model.value;

import java.io.File;
import java.util.Arrays;
import java.util.List;

/**
 * Abstract container for enums and constants associated with uploads.
 */
public abstract class Upload {
```

```
    public static abstract class Dirs {
        public static String RESOURCE_HTTP_PATH = "/uploads/**";
        public static String RESOURCES_BASE = "uploads";

        public static String IMAGES = RESOURCES_BASE + File.separator.concat("images").concat(File.separator);
        public static String VIDEOS = RESOURCES_BASE + File.separator.concat("videos").concat(File.separator);
        public static String PDFS = RESOURCES_BASE + File.separator.concat("pdfs").concat(File.separator);
    }

    public static abstract class SupportedFormats {
        private static String JPG = "jpg";
        private static String PNG = "png";
        private static String MP4 = "mp4";
        private static String PDF = "pdf";

        public static List<String> getSupportedFormats() {
            return Arrays.asList(JPG, PNG, MP4, PDF);
        }
    }

    public enum Type {
        IMAGE, VIDEO, PDF
    }
}
```

Repository

UploadModelRepository

```
package com.fatowlstudio.upload.repository;

import com.fatowlstudio.upload.model.UploadModel;
```

```
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;

import java.time.ZonedDateTime;
import java.util.List;

/**
 * Repository responsible for operations on {@link UploadModel}.
 */
public interface UploadModelRepository extends JpaRepository<UploadModel, Long> {

    /**
     * Retrieves list of unused upload models. Upload model is considered as unused if a file connected with this model
     * wasn't confirmed and it was uploaded before a date passed as a parameter.
     *
     * @return list of unused models
     */
    @Query("SELECT um FROM UploadModel um WHERE um.confirmed = false AND um.uploadDate < :limit")
    List<UploadModel> findUnusedModels(@Param("limit") ZonedDateTime limit);

    /**
     * Retrieves upload model with http resource link passed as a http link parameter.
     *
     * @param httpLink http resource link to search for
     * @return upload model with passed resource link
     */
    @Query("SELECT um FROM UploadModel um WHERE um.resourceHttpLink = :httpLink")
    UploadModel findByHttpResourceLink(@Param("httpLink") String httpLink);
}
```

Service

Impl

UploadModelServiceImpl

```
package com.fatowlstudio.upload.service.impl;

import com.fatowlstudio.upload.model.UploadModel;
import com.fatowlstudio.upload.repository.UploadModelRepository;
import com.fatowlstudio.upload.service.UploadModelService;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import java.time.ZoneOffset;
import java.time.ZonedDateTime;
import java.time.temporal.ChronoUnit;
import java.util.List;

/**
 * Default implementation of the {@link UploadModelService}. See it for public API documentation.
 */
@Service
@Transactional
public class UploadModelServiceImpl implements UploadModelService {

    private final UploadModelRepository uploadModelRepository;

    /**
     * Standard constructor.
     *
     * @param uploadModelRepository autowired UploadModelRepository
     */
    public UploadModelServiceImpl(UploadModelRepository uploadModelRepository) {
        this.uploadModelRepository = uploadModelRepository;
```

```
    }

    @Override
    public UploadModel createUploadModel(UploadModel uploadModelData) {
        return uploadModelRepository.save(uploadModelData);
    }

    @Override
    public UploadModel findByHttpResourceLink(String httpLink) {
        return uploadModelRepository.findByHttpResourceLink(httpLink);
    }

    @Override
    public List<UploadModel> findUnusedUploadModels() {
        return
uploadModelRepository.findUnusedModels(ZonedDateTime.now(ZoneOffset.UTC).minus(2L,
ChronoUnit.DAYS));
    }

    @Override
    public void remove(long modelId) {
        uploadModelRepository.delete(modelId);
    }

    @Override
    public void confirm(long modelId) {
        confirm(uploadModelRepository.findOne(modelId));
    }

    @Override
    public void confirm(String httpLink) {
        confirm(findByHttpResourceLink(httpLink));
    }

    private void confirm(UploadModel toConfirm) {
```

```
        toConfirm.setConfirmed(true);
        uploadModelRepository.save(toConfirm);
    }
}
```

UploadServiceImpl

```
package com.fatowlstudio.upload.service.impl;

import com.fatowlstudio.core.utils.CodingUtils;
import com.fatowlstudio.upload.model.UploadModel;
import com.fatowlstudio.upload.model.value.Upload;
import com.fatowlstudio.upload.service.UploadModelService;
import com.fatowlstudio.upload.service.UploadService;
import com.fatowlstudio.upload.utils.FileUploadUtils;
import com.fatowlstudio.upload.utils.FileUploadValidator;
import com.fatowlstudio.upload.utils.IOUploadUtils;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import org.springframework.web.multipart.MultipartFile;

import java.io.File;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;

/**
 * Default implementation for {@link UploadService} interface. See the interface for public API documentation.
 */
@Service
@Transactional
public class UploadServiceImpl implements UploadService {
```

```
@Value("${app.host}")
private String appHost;

@Value("${app.ssl}")
private boolean appSSL;

private final UploadModelService uploadModelService;

/**
 * Standard constructor.
 *
 * @param uploadModelService autowired UploadModelService
 */
public UploadServiceImpl(UploadModelService uploadModelService) {
   this.uploadModelService = uploadModelService;
}

@Override
public UploadModel uploadFile(MultipartFile file) throws IOException {

   FileUploadValidator.validate(file);

   Upload.Type uploadType =       FileUploadUtils.determineFileUploadType(file);
   String fileFormat =            FileUploadUtils.getFileExtension(file);
   String fileName =              CodingUtils.generateShortUUID().concat(".").concat(fileFormat);

   String targetDirectoryPath =    FileUploadUtils.determineFileTargetDirectory(uploadType);
   File targetDirectory =          IOUploadUtils.createDirectoryToWrite(targetDirectoryPath);
   File targetFile =              IOUploadUtils.createFileToWrite(targetDirectory, fileName);

   String  httpPath  =                    FileUploadUtils.buildHttpPathToResource(targetDirectory,
fileName, appHost, appSSL);
```

```
        IOUploadUtils.writeFile(targetFile, file);
        return uploadModelService.createUploadModel(
            new UploadModel(
                httpPath, targetFile.getAbsolutePath(), uploadType, fileFormat, targetFile.length() / 1024L, fileName
            )
        );
    }

    @Override
    public void deleteFile(String path) {
        try {
            Files.deleteIfExists(Paths.get(path));
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    @Override
    public void deleteFile(UploadModel uploadModel) {
        deleteFile(uploadModel.getResourcePhysicalLocation());
    }

    @Override
    public void deleteFileWithModel(UploadModel uploadModel) {
        uploadModelService.remove(uploadModel.getId());
        deleteFile(uploadModel.getResourcePhysicalLocation());
    }
}
```

UploadModelService

```
package com.fatowlstudio.upload.service;
```

```
import com.fatowlstudio.upload.model.UploadModel;

import java.util.List;

/**
 * Service responsible for operations on {@link UploadModel} entities.
 */
public interface UploadModelService {

    /**
     * Creates in the database an upload model passed as an upload model data parameter.
     *
     * @param uploadModelData upload model to be persisted
     * @return representation of the persisted upload model
     */
    UploadModel createUploadModel(UploadModel uploadModelData);

    /**
     * Returns upload model corresponding to a http resource link passed as a parameter.
     *
     * @param httpLink http link to be searched for
     * @return upload model with requested http link
     */
    UploadModel findByHttpResourceLink(String httpLink);

    /**
     * Returns list of unused upload models. Upload model is considered to be unused if its older that two days and
     * still wasn't confirmed.
     *
     * @return list of unused uploads
     */
    List<UploadModel> findUnusedUploadModels();
```

```
    /**
     * Deletes upload model with passed id.
     *
     * @param modelId id of the model to delete
     */
    void remove(long modelId);

    /**
     * Sets upload model with passed id to confirmed state.
     *
     * @param modelId id of the model to be set to confirmed
     */
    void confirm(long modelId);

    /**
     * Sets upload model with corresponding httpLink to confirmed state.
     *
     * @param httpLink httpLink of the model to be set to confirmed
     */
    void confirm(String httpLink);
}
```

UploadService

```
package com.fatowlstudio.upload.service;

import com.fatowlstudio.upload.model.UploadModel;
import com.fatowlstudio.upload.model.value.Upload;
import org.springframework.web.multipart.MultipartFile;

import java.io.IOException;

/**
 * Service responsible for uploading and managing files on the server.
```

```
 */
public interface UploadService {

    /**
     * Uploads multipart file to the server. Internal file location in the host file system will be decided based on
     * uploaded file type - by default jpg's nad png's are uploaded to the uploads/images folder, mp4's are uploaded to
     * the uploads/videos folder and pdf's are uploaded to the uploads/pdfs folder. Other files are not supported by
     * default. Those default paths can be changed by modifying {@link Upload.Dirs} class.
     *
     * An entity of the uploaded file will be created in the {@link UploadModel} - it is used to purge unused files and
     * to store file locations and other file metadata.
     *
     * @param file file to be uploaded
     * @return uploaded file model
     * @throws IOException in case of IO processing error
     */
    UploadModel uploadFile(MultipartFile file) throws IOException;

    /**
     * Physically removes file with provided file path from filesystem.
     *
     * @param path path of file to be removed
     */
    void deleteFile(String path);

    /**
     * Physically removes file bind with provided upload model from filesystem.
     *
     * @param uploadModel upload model of file to be removed
     */
    void deleteFile(UploadModel uploadModel);
```

```
    /**
     * Physically removes file bind with provided upload model and removes that upload model.
     *
     * @param uploadModel upload model to be removed with it's file
     */
    void deleteFileWithModel(UploadModel uploadModel);
}
```

Task

OldFilePurger

```
package com.fatowlstudio.upload.task;

import com.fatowlstudio.upload.service.UploadModelService;
import com.fatowlstudio.upload.service.UploadService;
import org.springframework.scheduling.annotation.Scheduled;
import org.springframework.stereotype.Component;
import org.springframework.transaction.annotation.Transactional;

/**
 * CRON job responsible for removing uploaded but unused files. It will delete each non-confirmed file from filesystem
 * and the database if it it's older than two days.
 */
@Component
public class OldFilePurger {

    private final UploadModelService uploadModelService;
    private final UploadService uploadService;

    public OldFilePurger(UploadModelService uploadModelService, UploadService uploadService) {
        this.uploadModelService = uploadModelService;
```

```
        this.uploadService = uploadService;
    }

    /**
     * Purges old, unused files every day at 1:00 a.m.
     */
    @Scheduled(cron = "0 0 1 * * *")
    @Transactional
    public void purgeOldFiles() {

uploadModelService.findUnusedUploadModels().forEach(uploadService::deleteFileWithModel);
    }
}
```

Utils

FileUploadUtils

```
package com.fatowlstudio.upload.utils;

import com.fatowlstudio.upload.config.PortListener;
import com.fatowlstudio.upload.model.value.Upload;
import org.apache.commons.io.FilenameUtils;
import org.springframework.web.multipart.MultipartFile;

import java.io.File;
import java.net.InetAddress;
import java.net.UnknownHostException;

/**
 * Abstract helper class providing many operations for {@link com.fatowlstudio.upload.service.impl.UploadServiceImpl} file upload
 * service implementation and others.
 */
public abstract class FileUploadUtils {
```

```
/**
 * Retrieves file extension from multipart file.
 *
 * @param file multipart file from which extension should be retrieved
 * @return extension of the provided file
 */
public static String getFileExtension(MultipartFile file) {
    return FilenameUtils.getExtension(file.getOriginalFilename()).toLowerCase();
}

/**
 * Discovers file upload type based on uploaded multipart file.
 *
 * @param file file of which upload type should be discovered
 * @return upload type of provided file
 */
public static Upload.Type determineFileUploadType(MultipartFile file) {
    switch (getFileExtension(file)) {
        case "jpg":
        case "png":
            return Upload.Type.IMAGE;
        case "pdf":
            return Upload.Type.PDF;
        case "mp4":
            return Upload.Type.VIDEO;
        default:
            throw new IllegalArgumentException("Unknown or not supported file type.");
    }
}

/**
 * Retrieves path to which file should be uploaded based on it's upload type.
 *
```

```
    * @param uploadType upload type of the file
    * @return path to which file should be uploaded
    */
   public static String determineFileTargetDirectory(Upload.Type uploadType) {
      switch (uploadType) {
         case IMAGE:
            return Upload.Dirs.IMAGES;
         case PDF:
            return Upload.Dirs.PDFS;
         case VIDEO:
            return Upload.Dirs.VIDEOS;
         default:
            throw new IllegalArgumentException("Unknown or not supported file type.");
      }
   }

   /**
    * Builds and returns http path to the uploaded file. It is the link at which file can be accessed by browsers.
    *
    * @param targetDirectory server directory at which file is stored
    * @param fileName filename
    * @return http link to the file
    * @throws UnknownHostException in case when application cannot specify host on which is running
    */
   public static String buildHttpPathToResource(File targetDirectory, String fileName, String httpHost, boolean isSSL) throws UnknownHostException {
      String protocol = isSSL ? "https://" : "http://";
      String httpHostProcessed = httpHost.equals("INET") ? InetAddress.getLocalHost().getHostAddress() : httpHost;
      String port = httpHost.equals("INET") ? ":".concat(Integer.toString(PortListener.runningPort)) : "";

      return protocol
            .concat(httpHostProcessed)
```

```
            .concat(port)
            .concat("/")
            .concat(targetDirectory.getPath().replace("\\", "/"))
            .concat("/")
            .concat(fileName);
    }
}
```

FileUploadValidator

```
package com.fatowlstudio.upload.utils;

import com.fatowlstudio.upload.ex.InvalidDataException;
import com.fatowlstudio.upload.model.value.Upload;
import org.springframework.web.multipart.MultipartFile;

/**
 * Validator for file uploads.
 */
public class FileUploadValidator {

    /**
     * Validates if file is properly prepared and valid. Exception will be thrown if not.
     *
     * @param file file to be validated
     */
    public static void validate(MultipartFile file) {
        if (!fileContainsData(file))
            throw new InvalidDataException("File cannot be empty.");
        if (!isFormatSupported(file))
            throw new InvalidDataException("Uploaded file type is not supported");
    }

    /**
```

```
    * Checks if file is not empty.
    *
    * @param file file to be checked
    * @return true if file is not empty, false otherwise
    */
   private static boolean fileContainsData(MultipartFile file) {
      return !file.isEmpty();
   }

   /**
    * Checks if uploaded file is one of supported formats.
    *
    * @param file file to be checked
    * @return true if file format is supported, false otherwise
    */
   private static boolean isFormatSupported(MultipartFile file) {
      return
Upload.SupportedFormats.getSupportedFormats().contains(FileUploadUtils.getFileExtension(file));
   }
}
```

IOUploadUtils

```
package com.fatowlstudio.upload.utils;

import org.springframework.web.multipart.MultipartFile;

import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;

/**
 * Helper class for performing IO operations while uploading files.
```

```
 */
public class IOUploadUtils {

    /**
     * Retrieves physical folder of target directory as a File object enabling all java.io file operations on it. If it
     * does not exist it will be created.
     *
     * @param targetDirectoryPath path of target directory
     * @return target directory as a File instance
     */
    public static File createDirectoryToWrite(String targetDirectoryPath) {
        File directory = new File(targetDirectoryPath);
        if (!directory.exists()) directory.mkdirs();
        return directory;
    }

    /**
     * Creates server file to which uploaded file can be written.
     *
     * @param targetDirectory directory in which file will be created
     * @param fileName name at which file will be created
     * @return empty server file ready to write
     */
    public static File createFileToWrite(File targetDirectory, String fileName) {
        File newServerFile = new File(
                targetDirectory.getAbsolutePath().concat(File.separator).concat(fileName)
        );

        if (newServerFile.exists())
            throw new IllegalStateException("Cannot upload file as it already exist on the server");

        return newServerFile;
    }
```

```
    /**
     * Physically writes uploaded file to the target file on server.
     *
     * @param targetFile file to which bytes should be written
     * @param uploadedFile bytes that should be written
     * @throws IOException in case of IO operations failure
     */
    public static void writeFile(File targetFile, MultipartFile uploadedFile) throws IOException {
        try (BufferedOutputStream stream = new BufferedOutputStream(new
FileOutputStream(targetFile))) {
            stream.write(uploadedFile.getBytes());
        }
    }
}
```

ANEKS

Narzędzie badawcze I: badanie kontekstowe

Badanie składało się z trzech części: 1) Historyjka – definicja pomocy, 2) Rutyna. Codzienne czynności. Trudności, 3) Sieci społeczne a pomoc.

1) Historyjka. Definicja pomocy

Pytanie 1.1.

Na początku mam dla Pani/Pana dwie historyjki, oto pierwsza z nich:

[***podajemy kartkę z historyjką na osobnej kartce!*** *Proszę te dwie historyjki skopiować z tego miejsca na osobne kartki. Jeżeli badani mają problem z czytaniem, to powoli, głośno i wyraźnie sami odczytujemy*]

Historyjka 1. W 10-piętrowym bloku zepsuła się winda. Pani Krystyna, która ma 78 lat, mieszka sama na siódmym piętrze i ma problemy z poruszaniem się, potrzebuje natomiast zrobić drobne zakupy spożywcze. Dzieci Pani Krystyny mieszkają w Anglii.

Pytanie 1.2. Co powinna zrobić Pani Krystyna? [*uwaga: po zakończeniu odpowiedzi przez badanych* ***dopytujemy o możliwości skorzystania z Internetu, z telefonu itp.]***

Pytanie 1.3. A teraz druga historyjka … [*wręczamy drugą historyjkę i prosimy o zapoznanie się z nią*]

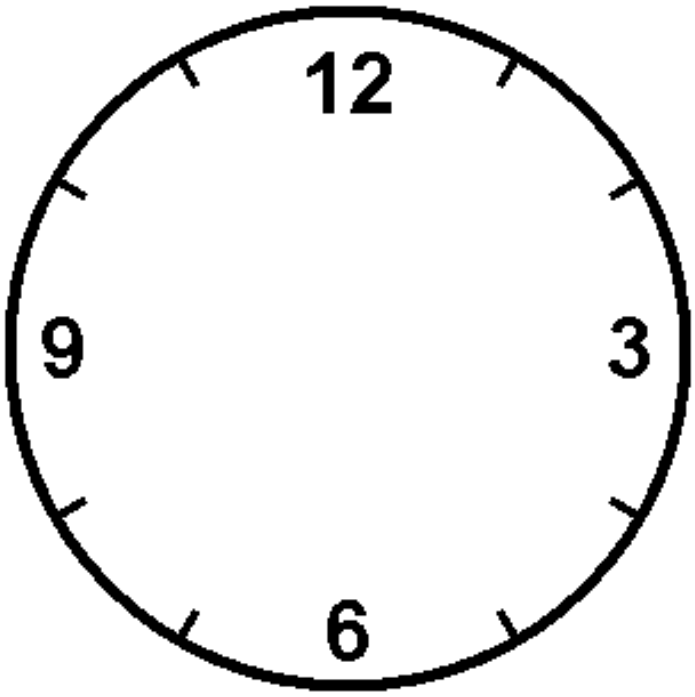
Historyjka 2. 73-letni Pan Zenon potknął się na chodniku i złamał nogę. Przez dwa miesiące jest unieruchomiony i nie może wychodzić z domu. Nie ma rodziny. Do obecnego mieszkania przeprowadził się dwa miesiące temu. Pan Zenon będzie potrzebował pomocy w załatwieniu różnych spraw poza domem.

Pytanie 1.4. Jak Pani/Pan myśli, w czym Pan Zenon może potrzebować pomocy? Co powinien zrobić?

2) *Rutyna. Czynności codzienne. Trudności*

Pytanie 2.1. [***UWAGA – ZEGAR DRUKUJEMY NA OSOBNEJ KARTCE***]

Chciał(a)bym teraz zapytać, jak wygląda Pani/Pana **zwykły dzień (poniedziałek-piątek)**. Mamy tutaj taki zegar i jakby Pan/Pani mogła opowiedzieć po kolei co Pani/Pan robi rano, przed południem, po południu oraz wieczorem i w nocy. Może Pan/Pani sam(a) zapisać hasłowo, albo ja w tym pomogę. Proszę do tego celu użyć koloru **czarnego** (długopis, marker). A czy **sobota** czymś się różni? Proszę dopisać [*albo robimy to sami*] sobotnie czynności kolorem **niebieskim**. A niedziela? Proszę dopisać [*albo robimy to sami*] niedzielne czynności kolorem **czerwonym**.



Pytanie 2.2. Czy któreś z tych czynności sprawiają Pani/Panu trudności? Dlaczego? Jak sobie Pani/Pan z tym radzi? A kto Pani/Panu pomaga? Jak to wygląda (czy sam prosi o pomoc? Czy ktoś stale sprawdza czy jest ok).

Pytanie 2.3. A może Pan/Pani w czymś komuś pomaga? Proszę o tym opowiedzieć.

Pytanie 2.4. A jaką rolę w pomaganiu odgrywają sąsiedzi? Jak wygląda z nimi kontakt? Czy sąsiedzi pomagają/pomagali Pani/Panu? A może Pani/Pan pomaga komuś z sąsiedztwa?

3) *Sieci społeczne a pomoc*

Pytanie 3.1.

W ramach ostatniej części tej rozmowy chciał(a)bym poznać osoby, na które może Pani/Pan liczyć.

[*W tym zadaniu należy podkreślić, iż wpisując poszczególne osoby nie trzeba wskazywać dokładnych danych osobowych. To mogą być inicjały*].

Pytania 3.2. Każdy z nas potrzebuje pomocy w bardziej lub mniej ważnych sprawach. Proszę wpisać w pierwszy okrąg do pięciu osób, które Pani/Panu pomagają w drobniejszych lub bardziej poważnych sprawach. [*W razie wątpliwości należy dodać, że mogą to być osoby z rodziny, ale też z pracownicy instytucji, wolontariusz*]

Pytanie 3.3. A teraz proszę w drugi okrąg wpisać do pięciu osób, do których zwróciłaby się Pani/Pan z prośbą o pomoc w sytuacji, gdy osoby wpisane do pierwszego okręgu nie mogą pomóc.

Pytanie 3.4. Kim są te osoby? [Pytamy kolejno o osoby P1, P2, P3 .. z kręgów o ich wiek, płeć, wykształcenie, skąd się znają, jak długo się znają, gdzie się najczęściej spotykają, jak często się spotykają, w jaki sposób się kontaktują ze sobą**].**

Pytanie 3.5. W czym pomagają te osoby *[Jeśli ktoś pomaga, to wówczas wstawiamy X w odpowiednim polu]?*

	P1=	P1=	P1=	P1=	P1=	P1=	P1=	P1=	P1=	P1=
W sprzątaniu										
W gotowaniu										
Robieniu prania										
Pomoc w sprawach urzędowych										
Doradzanie										
Wsparcie emocjonalne										
Robienie zakupów										
Zrealizowanie recepty										
Transport np. do lekarza, do urzędu, do koleżanki										
Nadzorowanie stanu zdrowia										
Pomoc w obsłudze komputera, telefonu, tabletu										

3.6. A może Pani/Pan komuś pomaga? Komu i jak?

	P1=	P1=	P1=	P1=	P1=	P1=	P1=	P1=	P1=	P1=
W sprzątaniu										
W gotowaniu										
Robieniu prania										
Pomoc w sprawach urzędowych										
Doradzanie										
Wsparcie emocjonalne										
Robienie zakupów										
Zrealizowanie recepty										
Transport np. do lekarza, do urzędu, do koleżanki										
Nadzorowanie stanu zdrowia										
Pomoc w obsłudze komputera, telefonu, tabletu										

Narzędzie badawcze II: test użyteczność aplikacji

Jak ocenia Pan/Pani wygląd aplikacji:

Proszę o ocenę wyglądu aplikacji w skali od 1(bardzo źle) do 5 (bardzo dobrze)

Kolory

	1	2	3	4	5	
bardzo źle	○	○	○	○	○	bardzo dobrze

Jeżeli kolory ocenił/a Pan/Pani negatywnie, to w jakich kolorach, Pana/Pani zdaniem, powinna być aplikacja

Tekst krótkiej odpowiedzi

Czcionka liter

	1	2	3	4	5	
bardzo źle	○	○	○	○	○	bardzo dobrze

Wielkość liter

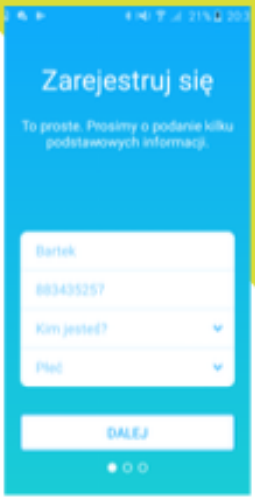
	1	2	3	4	5	
bardzo źle	○	○	○	○	○	bardzo dobrze

Symbole (ikony) określające funkcję

	1	2	3	4	5	
bardzo źle	○	○	○	○	○	bardzo dobrze

Pana/Pani uwagi dotyczące wyglądu aplikacji:

REJESTRACJA



Jak ocenia Pan/Pani proces rejestracji

Proszę o ocenę procesu rejestracji w skali od 1(bardzo źle) do 5 (bardzo dobrze)

Czy rejestracja powiodła się?

- ◯ Tak
- ◯ Nie

Jeśli w poprzednim pytaniu odpowiedź brzmi "Nie" to proszę opisać problem jaki wystąpił:

Tekst krótkiej odpowiedzi

Proces rejestracji

	1	2	3	4	5	
bardzo źle	◯	◯	◯	◯	◯	bardzo dobrze

Intuicyjność

	1	2	3	4	5	
bardzo źle	◯	◯	◯	◯	◯	bardzo dobrze

Czytelność

	1	2	3	4	5	
bardzo źle	◯	◯	◯	◯	◯	bardzo dobrze

Pana/Pani uwagi dotyczące rejestracji aplikacji:

LOGOWANIE

Jak ocenia Pan/Pani proces logowania

Proszę o ocenę procesu logowania w skali od 1(bardzo źle) do 5 (bardzo dobrze)

Czy logowanie powiodło się?

- ◯ Tak
- ◯ Nie

Czy w czasie logowania nastąpił problem i nie udało się zalogować do aplikacji?

- ◯ Tak
- ◯ Nie

Proces logowania był dla mnie:

	1	2	3	4	5	
bardzo trudny	◯	◯	◯	◯	◯	bardzo łatwy

Logowanie do aplikacji oceniam

	1	2	3	4	5	
bardzo źle	◯	◯	◯	◯	◯	bardzo dobrze

Pana/Pani uwagi dotyczące logowania się do aplikacji:

Udzielanie Pomocy

Proszę o ocenę procesu logowania w skali od 1(bardzo źle) do 5 (bardzo dobrze)

Czy udzielił/a Pan/Pani pomocy osobie potrzebującej?

- ◯ Tak
- ◯ Nie
- ◯ Nie dotyczy

Jeśli odpowiedź brzmi "Tak" to proszę wskazać ile razy takiej pomocy Pan/Pani udzielił/a?

Tekst krótkiej odpowiedzi

Czy komunikaty wysyłane przez aplikację były czytelne?

- ◯ Tak
- ◯ Nie
- ◯ Nie dotyczy

Pana/Pani uwagi dotyczące opcji "Pomagam" w aplikacji:

Korzystanie z Pomocy

Proszę o ocenę procesu logowania w skali od 1(bardzo źle) do 5 (bardzo dobrze)

Czy potrzebował/a Pan/Pani skorzystać z pomocy w kategorii ZAKUPY?

- ◯ Tak
- ◯ Nie
- ◯ Nie dotyczy

Jeśli odpowiedź brzmi "Tak" to proszę ocenić tą funkcjonalność:

	1	2	3	4	5	
bardzo źle	◯	◯	◯	◯	◯	bardzo dobrze

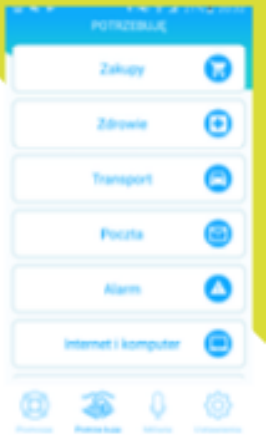
Czy potrzebował/a Pan/Pani skorzystać z pomocy w kategorii ZDROWIE?

- ◯ Tak
- ◯ Nie

Jeśli odpowiedź brzmi "Tak" to proszę ocenić tą funkcjonalność:

	1	2	3	4	5	
bardzo źle	◯	◯	◯	◯	◯	bardzo dobrze

KORZYSTANIE POMOCY



Czy potrzebował/a Pan/Pani skorzystać z pomocy w kategorii TRANSPORT?

- Tak
- Nie

Jeśli odpowiedź brzmi "Tak" to proszę ocenić tą funkcjonalność:

	1	2	3	4	5	
bardzo źle	○	○	○	○	○	bardzo dobrze

Czy potrzebował/a Pan/Pani skorzystać z pomocy w kategorii POCZTA?

- Tak
- Nie

Jeśli odpowiedź brzmi "Tak" to proszę ocenić tą funkcjonalność:

	1	2	3	4	5	
bardzo źle	○	○	○	○	○	bardzo dobrze

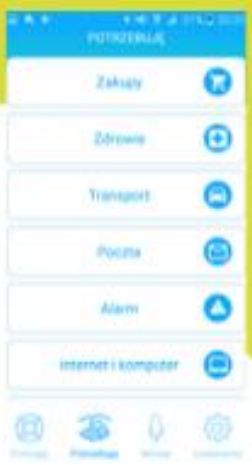
Czy potrzebował/a Pan/Pani skorzystać z pomocy w kategorii ALARM?

- Tak
- Nie

Jeśli odpowiedź brzmi "Tak" to proszę ocenić tą funkcjonalność:

	1	2	3	4	5	
bardzo źle	○	○	○	○	○	bardzo dobrze

KORZYSTANIE POMOCY



Czy potrzebował/a Pan/Pani skorzystać z pomocy w kategorii INTERNET I KOMPUTER?

- Tak
- Nie

Jeśli odpowiedź brzmi "Tak" to proszę ocenić tą funkcjonalność:

	1	2	3	4	5	
bardzo źle	○	○	○	○	○	bardzo dobrze

Czy uważa Pan/Pani, że należy dodać nowe kategorie do wyboru?

- Tak
- Nie

Jeśli twoja odpowiedź brzmi "Tak" to proszę zaproponuj swoje nowe kategorie:

Tekst krótkiej odpowiedzi

Czy uważa Pan/Pani, że któraś z kategorii jest zbędna?

- Tak
- Nie

Jeśli Pana/Pani odpowiedź brzmi "Tak" to proszę wskaż zbędne kategorie:

Tekst krótkiej odpowiedzi

Pana/Pani uwagi dotyczące opcji "Potrzebuję" w aplikacji:

NAGRYWANIE GŁOSOWE

Korzystanie z funkcji głosowej

Proszę o ocenę trybu głosowego w skali od 1(bardzo źle) do 5 (bardzo dobrze)

Czy korzystał/a Pan/Pani z trybu głosowego?

- ◯ Tak
- ◯ Nie

Jeśli Pana/Pani odpowiedź brzmi "Tak" to jak oceniasz tą funkcjonalność:

	1	2	3	4	5	
bardzo źle	◯	◯	◯	◯	◯	bardzo dobrze

Pana/Pani uwagi dotyczące funkcji głosowej aplikacji:

Dokonywanie zmian w ustawieniach

Proszę o ocenę ustawień aplikacji w skali od 1(bardzo źle) do 5 (bardzo dobrze)

Czy dokonywał/a Pan/Pani zmian w ustawieniach aplikacji

- ◯ Tak
- ◯ Nie

Jeśli Pana/Pani odpowiedź brzmi "Tak" to jak oceniasz tą funkcjonalność:

	1	2	3	4	5	
bardzo źle	◯	◯	◯	◯	◯	bardzo dobrze

Pana/Pani uwagi dotyczące ustawień aplikacji:

DZIENNICZEK

W tym miejscu proszę o rejestrację problemów pojawiających się w czasie użytkowania aplikacji

DATA	PROBLEM

Źródła danych statystycznych

1. Główny Urząd Statystyczny
2. Diagnoza Społeczna
3. CBOS
4.

Wykorzytane grafiki:

Profil 1#1 zdjęcie: Cesar Vargas, 2009, Vovu, Flickr.com, icencja Creative Commons, http://bit.ly/1SeWFHk

Profil #2 zdjęcie: Nicolas Alejandro, 2014, Street artist, Flickr.com, licencja Creative Commons, http://bit.ly/1kiUfvr

Profil #3 zdjęcie: Makelessnoise, 2007, Grand Lady http://bit.ly/1KNv89y

Licencja Creative Commons, http://www.freepik.com